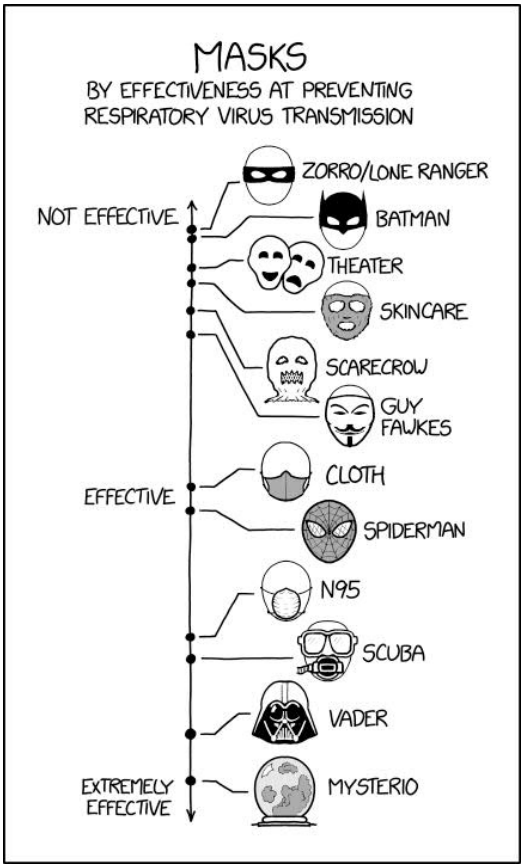
Tampereen yliopisto
Tampere University

COMP.SE.100-EN ItSE

Zoom begins soon...

at 1415 o'clock.



Tampereen yliopisto
Tampere University

COMP.SE.100-EN, 2020, course schedule v6e (09.11.2020)

week	lectures	exam	weekly exercises	project assignment (exercise work)	week
35	L1: course basics		--- sign to WE groups ---	sign for project = grouping...	35
36	Project Assignment explained		**WE1**: intro to requirements	grouping, groups to Moodle	36
37	L2: Sw Eng in general		**WE2**: Trello and agile way	group's Trello board ready with product backlog	37
38	L3: requirements		**WE3**: feasibility study and stakeholder analysis	working...	38
39	L4: basic UML diagrams		**WE4**: requirements	working...	39
40	L5: more UML diagrams	EXAM-1	**WE5**: UML diagrams - Use case	working...	40
41	L6: different sw systems	EXAM-1	**WE6**: UML diagrams - concept/entity and navigatio	deadline for 1st phase documentation and presentation	41
42	examination week		examination week	examination week	42
43	L7: life cycle models		groups' 1st presentations	groups' 1st phase presentations	43
44	L8: quality and testing		**WE7**: development processes	feedback group-to-group at PRP, from 1st phase	44
45	L9: project work	Forms-2	**WE8**: testing and error reporting	deadline for diagrams first versions (Moodle)	45
46	L10: project management		**WE9**: effort estimation	feedback to groups from diagrams (from assistants)	46
47	L11: open source, APIs, IPR		**WE10**: delivery contracts and terms of use	deadline for 2nd phase presentation (PRP)	47
48	L12: embedded systems, IoT		groups' final presentations	groups' final presentations / feedback g-to-g (PRP)	48
49	L13: recap, summary	Forms-3	- - -	final (2.) delivery of project documentation	49
50	examination week		examination week	feedback inside group, student-to-student at PRP	50
51	examination week		examination week	`end of game / game over.`	51
	Lectures: Wed at 1415-16.		Weekly exercises:		
			Mon 0815-10 discontinued	AUTUMN 2020 (1-2. periods)	
			Mon 1215-14	are remote/distant learning.	
			Tue 0815-10		
			Tue 1415-16		
			Wed 0815-10 discontinued .		

Remote/distant learning 2020.
No contact teaching at ItSE 2020.

Tampereen yliopisto
Tampere University

COMP.SE.100 -EN "ItSE"

Introduction to Software Engineering

2020, 1-2. periods

5 credit units

10-projman-ItSE-2020-v6

Tampereen yliopisto
Tampere University

COMP.SE.100-EN (ItSE)
Introduction to Software Engneering

Lecture 10, 11.11.2020

Tensu: remember to start Zoom lecture recording, at 1415

Prefer course Moodle over SISU information.

Students are recommended to follow Moodle News/messages.

Tampereen yliopisto
Tampere University

COMP.SE.100-EN, 2020, course schedule v6e (09.11.2020)

week	lectures	exam	weekly exercises	project assignment (exercise work)	week
35	L1: course basics		--- sign to WE groups ---	sign for project = grouping...	35
36	Project Assignment explained		**WE1**: intro to requirements	grouping, groups to Moodle	36
37	L2: Sw Eng in general		**WE2**: Trello and agile way	group's Trello board ready with product backlog	37
38	L3: requirements		**WE3**: feasibility study and stakeholder analysis	working...	38
39	L4: basic UML diagrams		**WE4**: requirements	working...	39
40	L5: more UML diagrams	EXAM-1	**WE5**: UML diagrams - Use case	working...	40
41	L6: different sw systems	EXAM-1	**WE6**: UML diagrams - concept/entity and navigatio	deadline for 1st phase documentation and presentation	41
42	examination week		examination week	examination week	42
43	L7: life cycle models		groups' 1st presentations	groups' 1st phase presentations	43
44	L8: quality and testing		**WE7**: development processes	feedback group-to-group at PRP, from 1st phase	44
45	L9: project work	Forms-2	**WE8**: testing and error reporting	deadline for diagrams first versions (Moodle)	45
46	L10: project management		**WE9**: effort estimation	feedback to groups from diagrams (from assistants)	46
47	L11: open source, APIs, IPR		**WE10**: delivery contracts and terms of use	deadline for 2nd phase presentation (PRP)	47
48	L12: embedded systems, IoT		groups' final presentations	groups' final presentations / feedback g-to-g (PRP)	48
49	L13: recap, summary	Forms-3	- - -	final (2.) delivery of project documentation	49
50	examination week		examination week	feedback inside group, student-to-student at PRP	50
51	examination week		examination week	`end of game / game over.`	51
	Lectures: Wed at 1415-16.		Weekly exercises:		
			Mon 0815-10 discontinued	AUTUMN 2020 (1-2. periods)	
			Mon 1215-14	are remote/distant learning.	
			Tue 0815-10		
			Tue 1415-16		
			Wed 0815-10 discontinued .		

Tampereen yliopisto
Tampere University

"Once I learn how to use Google, isn't that all the education I really need?"

Tampereen yliopisto
Tampere University

Current at course (w 46)

- WE9 were this week (estimations)
- next week the last WE10

- continue updating your Trello (kanban) boards = use at your process

- deadline for diagrams' first versions, to Moodle, last week 45

- Third exam is changed to Forms, as covid is still here.

- EXAM 3/3 w48-49 is changed to Forms-3 exam, to be on Wednesday, 02.12.2020 starting at 1615 o'clock.

Tampereen yliopisto
Tampere University

First, general course matters

Project assignment (exercise work)

Juanita: groups G01-G04

Aleksius: ODD groups; G05,G07,G09,G11,G13,G15,G17,G19,G21,G23,G27

Lauri: EVEN groups; G06,G08,G10,G12,G14,G16,G18,G20,G22,G24,G28

- Trello board is used as help for work division and assignment

WE attendees:

- Mon 0815-10 9, 8,10, 5, 6, 4,
- **Mon 1215-14 11,12,12,13,11,12,10, 9, 9,**
- **Tue 0815-10 3, 6, 4, 6, 5, 5,10,12,10,**
- **Tue 1415-16 8,10, 9, 8, 5, 7,12,12,12,**
- Wed 0815-10 12,11, 9, 8, 7, 6,

Very small WEs are not reasonable, we discontinued two WE groups at 2nd study period.

Tampereen yliopisto
Tampere University

Course contents (plan)

1. Course basics, intro
2. Sw Eng in general, overview
3. Requirements
4. Different software systems
5. Basic UML Diagrams (”Class”, Use Case, Navigation)
6. Life Cycle models
7. UML diagrams, in more detail
8. Quality and Testing
9. Project work
10. **Project management**
11. Open source, APIs, IPR
12. Embedded systems
13. Recap

Tampereen yliopisto
Tampere University

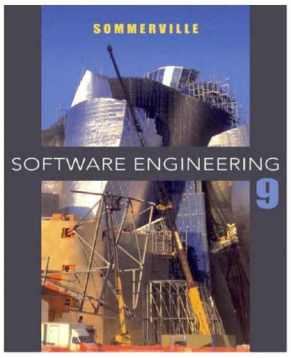
10. Project management

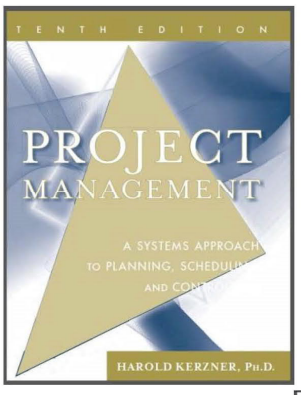
- overview of a project
- project management
- common problems (How to Fail)
- common success factors (Best Practices, Patterns)
- project management skills
- tools
- scheduling / time estimations
- metrics / measurements (complexity, FPA, FiSMA, COCOMO,...)
- change management
- risk management
- indicators (e.g. ”traffic lights”)
- standards

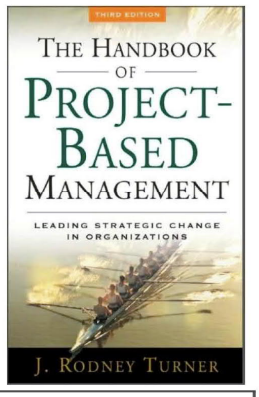
Overview of a project management

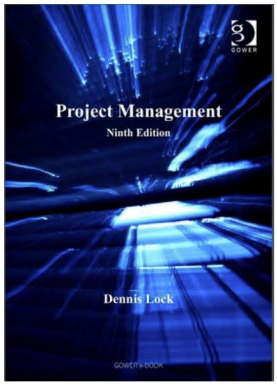
Some famous PM (project management) books at top row.

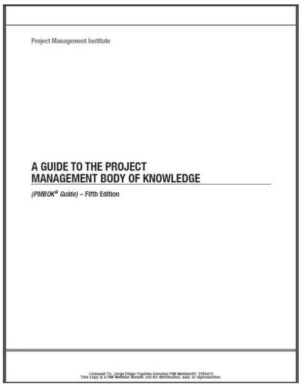
I think at least 50 % of PM matters are valid for all application areas, so not depending which business or technical branch projects you have.



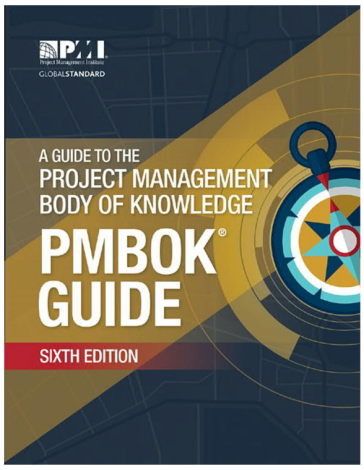








41



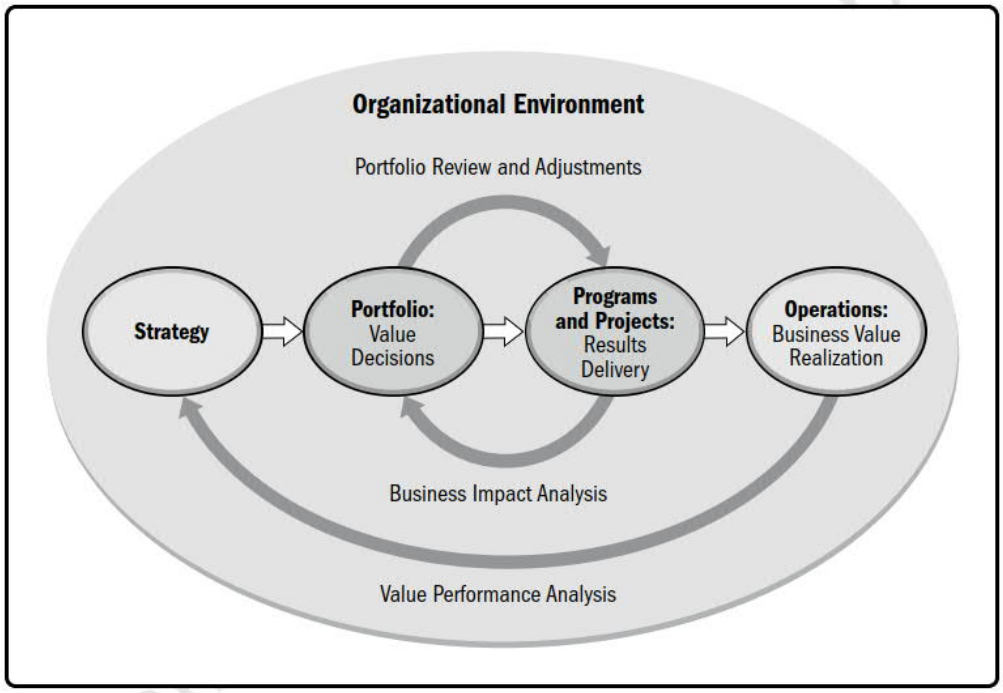


Figure 1-4. Organizational Project Management

The big picture:
Portfolio
Programs
Projects.

Tavanomainen sanasto,
mutta ei aina näin:

portfolio = projektisalkku
program = ohjelma, hanke
project = projekti

Toisaalta (Virkki-Somermeri)
ohjelma
hanke
projekti.

SFS: portfolio = salkku
programme = ohjelma
project = projekti
projekti = tilapäinen hanke (temporary endeavour)

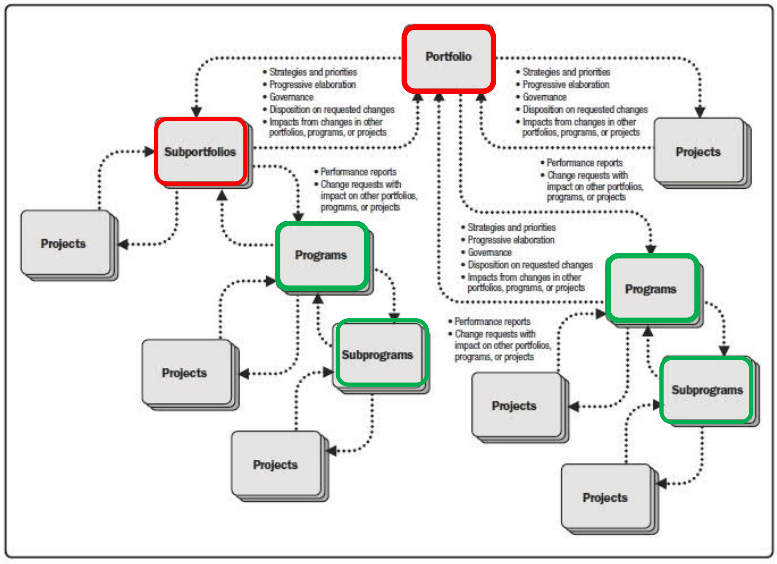
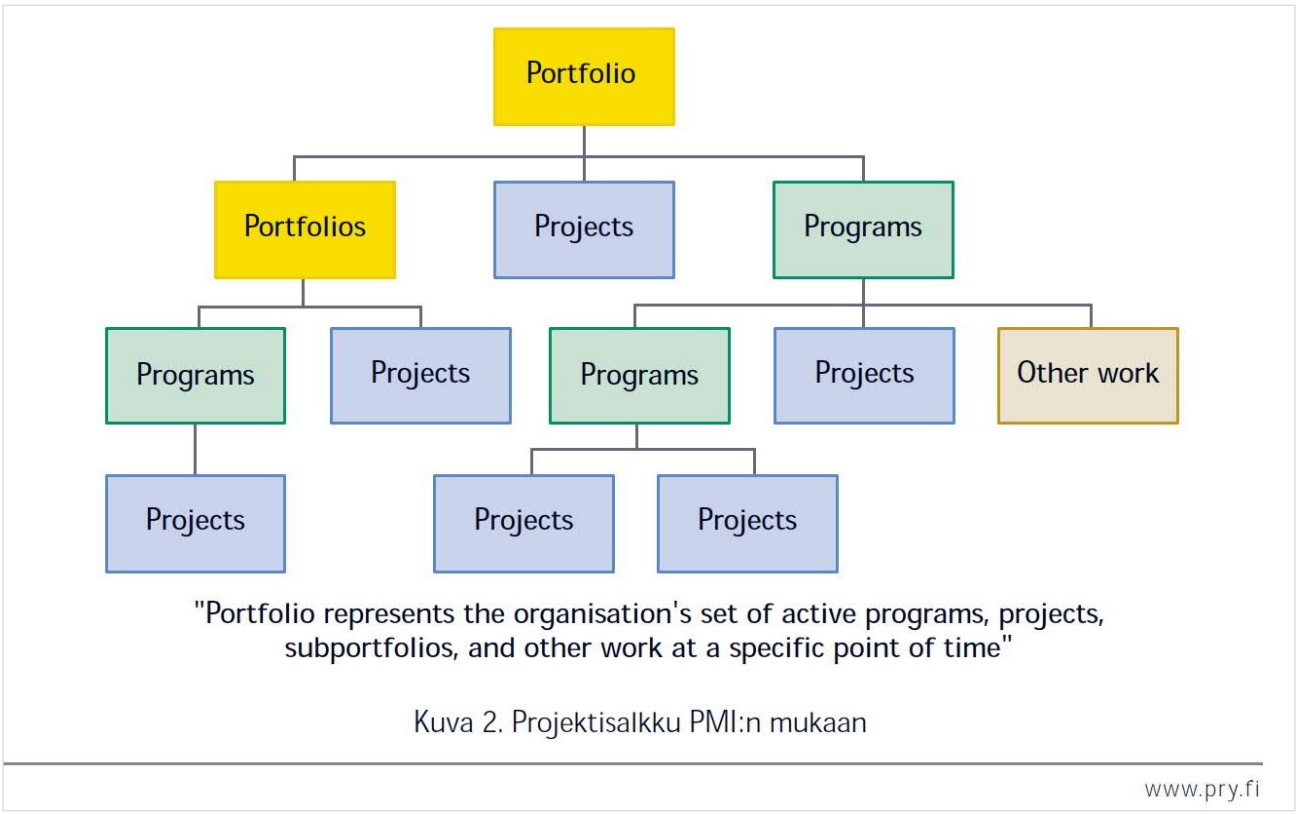
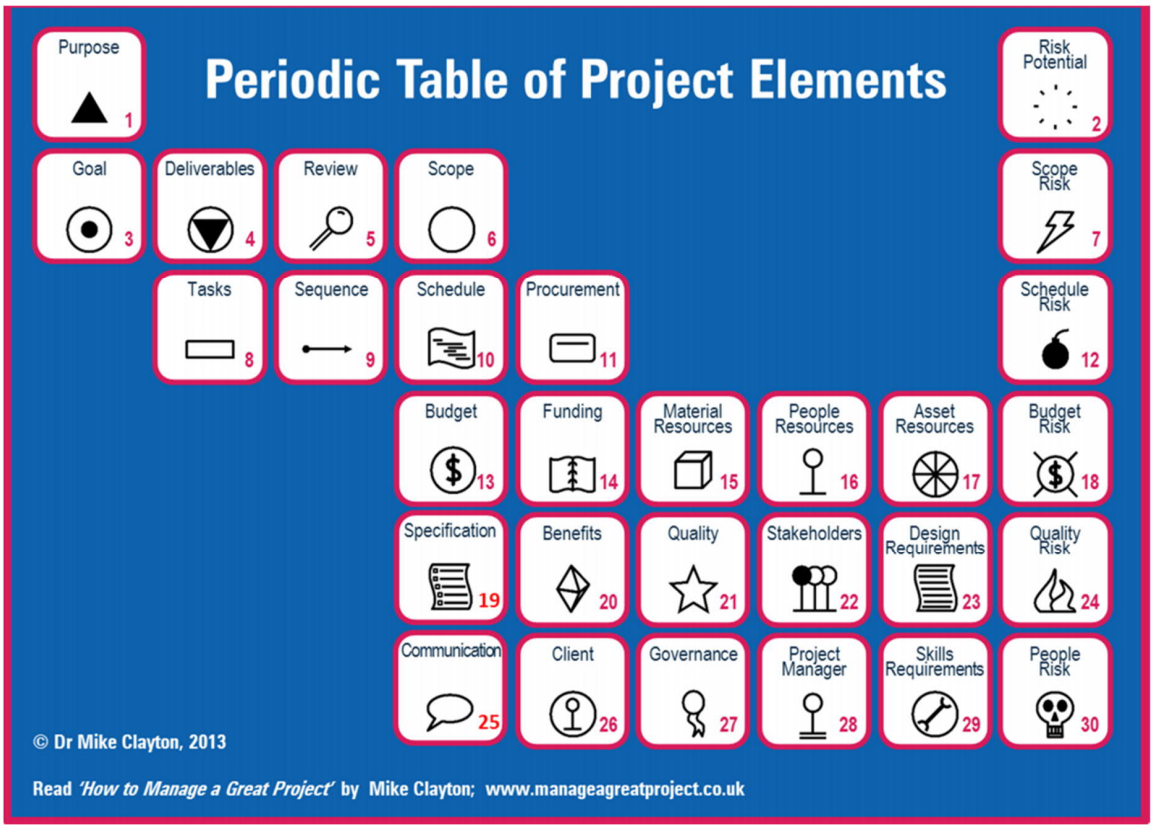


Figure 1-1. Portfolio, Program, and Project Management Interactions

[Guide to PMBoK, 5th ed, 2013]



A project manager have better take these parts into account.



Project documentation checklist; project plan and possible project documents

Don't be scared, this is for heavy/big projects.

But it is a good checklist that you forget nothing.

Table 4-1. Project Management Plan and Project Documents

Project Management Plan	Project Documents	
1. Scope management plan	1. Activity attributes	19. Quality control measurements
2. Requirements management plan	2. Activity list	20. Quality metrics
3. Schedule management plan	3. Assumption log	21. Quality report
4. Cost management plan	4. Basis of estimates	22. Requirements documentation
5. Quality management plan	5. Change log	23. Requirements traceability matrix
6. Resource management plan	6. Cost estimates	24. Resource breakdown structure
7. Communications management plan	7. Cost forecasts	25. Resource calendars
8. Risk management plan	8. Duration estimates	26. Resource requirements
9. Procurement management plan	9. Issue log	27. Risk register
10. Stakeholder engagement plan	10. Lessons learned register	28. Risk report
11. Change management plan	11. Milestone list	29. Schedule data
12. Configuration management plan	12. Physical resource assignments	30. Schedule forecasts
13. Scope baseline	13. Project calendars	31. Stakeholder register
14. Schedule baseline	14. Project communications	32. Team charter
15. Cost baseline	15. Project schedule	33. Test and evaluation documents
16. Performance measurement baseline	16. Project schedule network diagram	
17. Project life cycle description	17. Project scope statement	
18. Development approach	18. Project team assignments	

[PMBOK Guide 2017]

[https://www.smartsheet.com/content/software-project-management]

Project managers should consider the following table as a guide to provide the best exposure of the information derived from their project plans.

User Type	Planning Details Needed	Delivery Tools
Project planners	Tracking and scheduling of: • Milestones • Activities • Dependencies • Multiple baselines • Critical path • Assignments • Timelines	Tools include: • Work Breakdown Structure (WBS) view • Gantt Charts • Documents • Dashboards and reports
Project workers	Tracking and updating of: • My work • Issues • Tasks • Assignments • To-dos • Documents	Tools include: • My work view • Task list view • Time sheet view • Kanban view of work
Project stakeholders	Tracking and sharing of: • Project and plan progress and status • Planned vs. actual of workload and costs • Risks, issues and changes	Tools include: • Multi-project and portfolio dashboards and views • Traffic light indicator views • Multi-project Gantt views • Workload and cost reporting and charts

BOSCARD [https://www.projectsmart.co.uk/boscard.php]

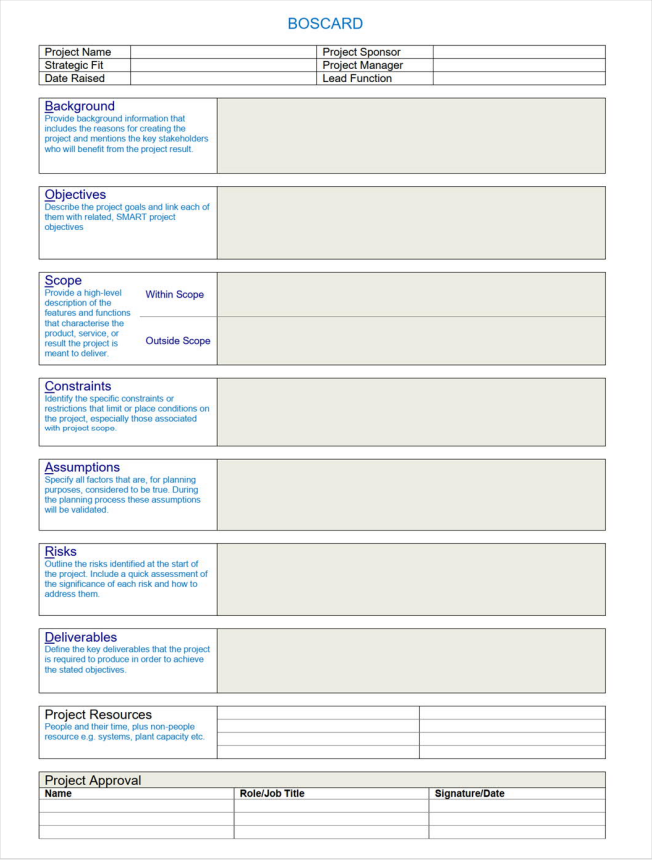
Think before you act, do not take shortcuts in the beginning of a new project

Background	Provide background information that includes the reasons for creating the project and mentions the key stakeholders who will benefit from the project result.
Objectives	Describe the project goals and link each of them with related, SMART project objectives.
Scope	Provide a high-level description of the features and functions that characterise the product, service, or result the project is meant to deliver.
Constraints	Identify the specific constraints or restrictions that limit or place conditions on the project, especially those associated with project scope.
Assumptions	Specify all factors that are, for planning purposes considered to be true. During the planning process, these assumptions will be validated.
Risks	Outline the risks identified at the start of the project. Include a quick assessment of the significance of each risk and how to deal with them.
Deliverables	Define the key deliverables the project is required to produce to achieve the stated objectives.

BOSCARD

The BOSCARD is a strategic planning tool used to give the terms-of-reference for new projects.

When initiating a project, it is important that all parties involved agree in considerable detail what the project is to achieve before it starts. Failure to gain formal agreement almost always leads to some expectations not being met.



BOSCARD

Project Name		Project Sponsor	
Strategic Fit		Project Manager	
Date Raised		Lead Function	

Background
Provide background information that includes the reasons for creating the project and mentions the key stakeholders who will benefit from the project result.

Objectives
Describe the project goals and link each of them with related, SMART project objectives

Scope
Provide a high-level description of the features and functions that characterise the product, service, or result the project is meant to deliver.
Within Scope
Outside Scope

Constraints
Identify the specific constraints or restrictions that limit or place conditions on the project, especially those associated with project scope.

Assumptions
Specify all factors that are, for planning purposes, considered to be true. During the planning process these assumptions will be validated.

Risks
Outline the risks identified at the start of the project. Include a quick assessment of the significance of each risk and how to address them.

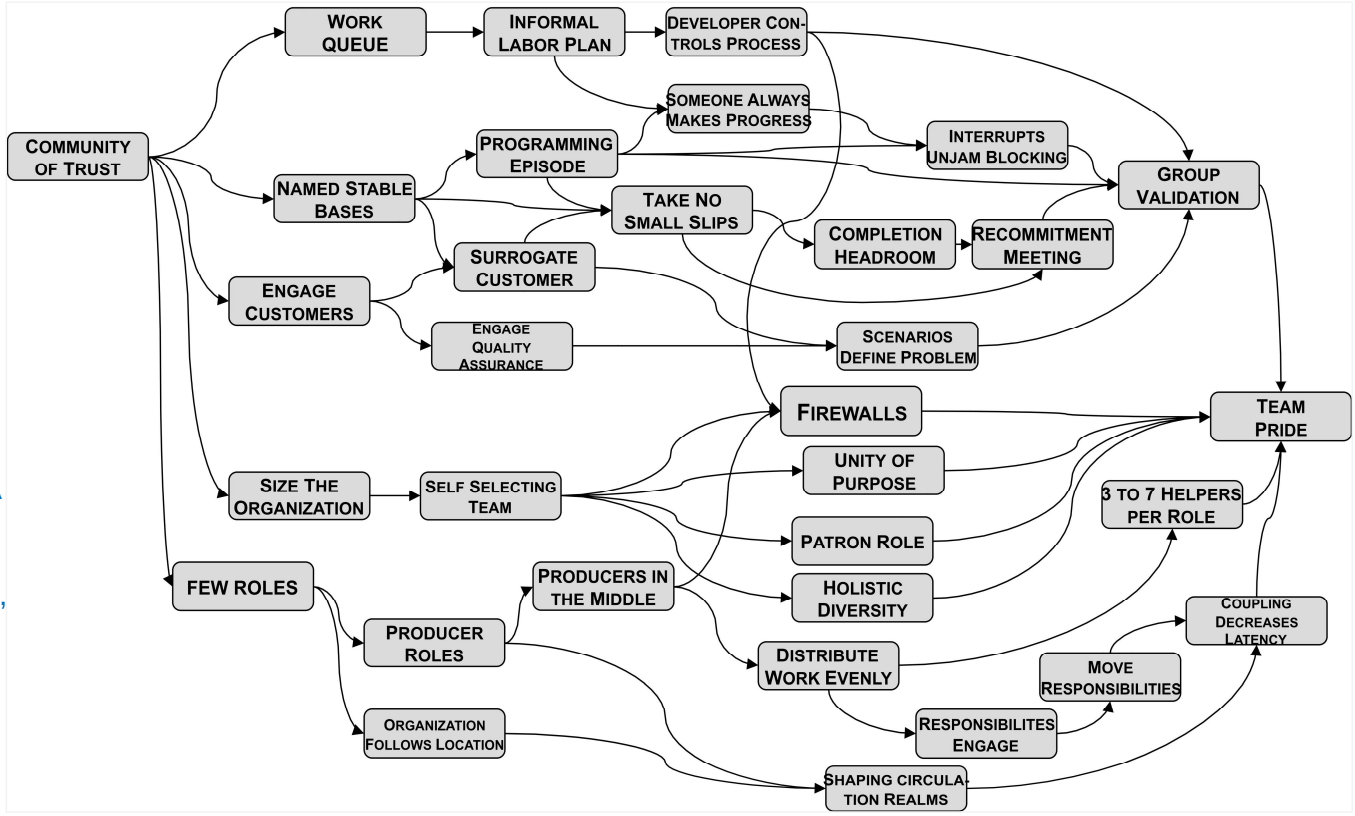
Deliverables
Define the key deliverables that the project is required to produce in order to achieve the stated objectives.

Project Resources
People and their time, plus non-people resource e.g. systems, plant capacity etc.

Project Approval

Name	Role/Job Title	Signature/Date

James Coplien: A Scrum Pattern Language, 2008



Common pitfalls and solutions to facilitate the quick and successful development of projects plans

1) Don't bite off more than you can chew.

Plans do not need to be fully developed before they are put into action. The reality is that the project plan as a living document will be adjusted with the ebb and flow of the project. Use Pareto's 80/20 rule to get the ball rolling. Then let your team and stakeholders guide the document as needed.

2) No need to start from scratch.

Reinventing the wheel will not necessarily make for a better plan. In fact, too much time is often wasted on building a plan that undergoes a number of changes down the line. Leveraging existing project templates and best practices is an excellent way to reduce the initial effort in project planning.

3) Avoid analysis paralysis.

An over-detailed plan, in many cases, results in an ineffective execution on granular activities and causes problems in the tracking progress. Finding that right balance of detail in the plan allows team members to have a better understanding in their execution while delivering the necessary project data for stakeholder decisions.

4) Don't build the plan alone.

One of the most critical steps in successful project planning is including stakeholder buy-in and involvement in the development process. The plan needs to be the roadmap of the entire project and all parties are responsible for its success. Working in a vacuum and building a plan based on assumptions is dangerous. The plan will always be imperfect and evolving. For this reason alone, stakeholder involvement is needed to help guide and approve the project plan despite the final results.

[The art of project planning, 2014]

RACI table (matrix, chart), responsibility assignment

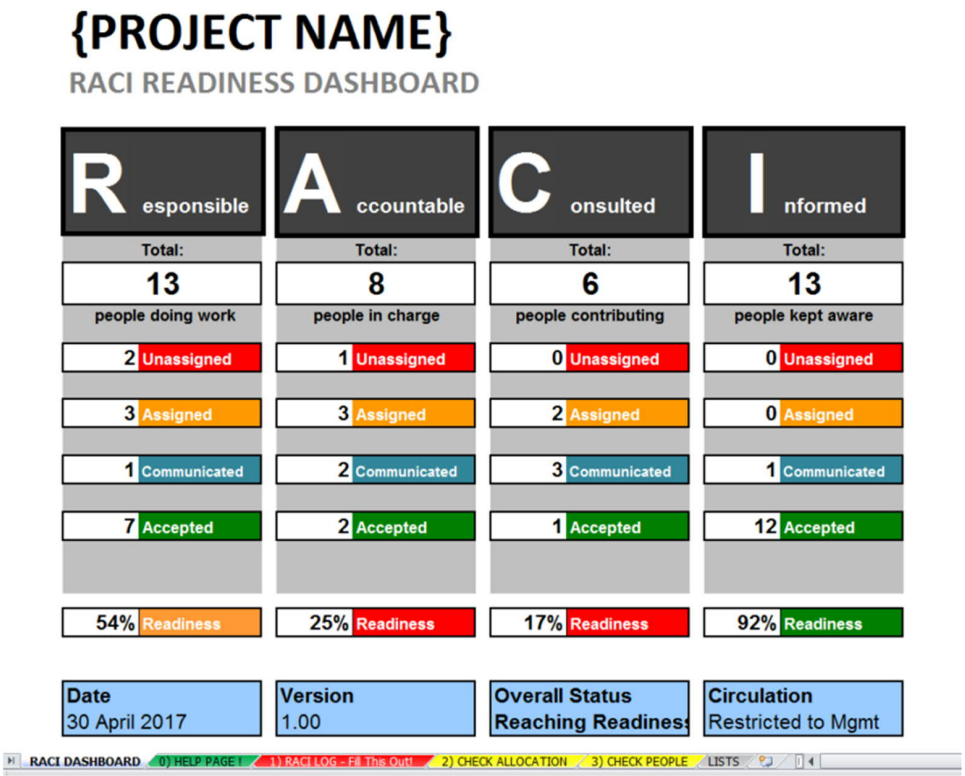
R = Responsible
Those who do work to achieve the activity, there can be multiple resources responsible.

A = Accountable
The role/resource who will sign off on the work and judge its completion and how it meets quality standards. There must be only one "A" specified for each activity.

C = Consulted
Those whose opinions, skills, knowledge are sought to complete the activity.

I = Informed
Those that need to know about the activity.

RACI, Responsibility assignment matrix



{PROJECT NAME}

RACI READINESS DASHBOARD

Responsible	Accountable	Consulted	Informed
Total:	Total:	Total:	Total:
13	8	6	13
people doing work	people in charge	people contributing	people kept aware
2 Unassigned	1 Unassigned	0 Unassigned	0 Unassigned
3 Assigned	3 Assigned	2 Assigned	0 Assigned
1 Communicated	2 Communicated	3 Communicated	1 Communicated
7 Accepted	2 Accepted	1 Accepted	12 Accepted
54% Readiness	25% Readiness	17% Readiness	92% Readiness

Date	Version	Overall Status	Circulation
30 April 2017	1.00	Reaching Readines	Restricted to Mgmt

RACI DASHBOARD | 0) HELP PAGE ! | 1) RACI LOG – Fill This Out! | 2) CHECK ALLOCATION | 3) CHECK PEOPLE | LISTS

[https://business-docs.co.uk/downloads/raci-template-managing-project-responsibilities-excel/]

Tampereen yliopisto
Tampere University

RACI

helpful when tasks require multiple resources, run concurrently, or depend on other tasks.

{PROJECT NAME}				
RACI LOG	The table below has placeholder content. Please delete (without deleting the headers) and add your tasks, and people for each task. See the "HELP PAGE" sheet for more info.			
TASK	Person	RACI Role	Status	Notes
Assemble the team	Sarah Beezwax	Responsible	Assigned	Sarah might not have time for this, but we need her
Assemble the team	Simon Tarnish	Accountable	Assigned	Simon is the only person - we need him to run this strand.
Assemble the team	Marvyn Cloomes	Consulted	Communicated	Got to get his input, otherwise the widgets will be at risk.
Assemble the team	Silov Gatestata	Informed	Accepted	Silov just needs to feel he's in the loop.
Find the budget		Responsible	Unassigned	Need someone to actually do this work
Find the budget	Jo Fennel	Accountable	Communicated	
Find the budget	Sammy Fraves	Consulted	Communicated	
Find the budget	Mr Yeates	Informed	Accepted	
Find the working space		Responsible	Unassigned	
Find the working space	Mrs T Sparketor	Accountable	Assigned	
Find the working space	Martin Crow	Consulted	Communicated	
Find the working space	Ellie Murphy	Responsible	Accepted	
Secure the working space	Uncle Nige	Responsible	Accepted	
Secure the working space	Jo Baby	Accountable	Accepted	
Secure the working space	Ellie Murphy	Consulted	Accepted	
Secure the working space	Niall Spartan	Informed	Accepted	
Establish the PROJECT SCOPE	Sammy Fraves	Accountable	Assigned	
Establish the PROJECT SCOPE	Mr Yeates	Responsible	Accepted	
Establish the PROJECT SCOPE	Hank Mercator	Responsible	Accepted	
Establish the PROJECT SCOPE	Martin Tucci	Responsible	Accepted	
Establish the PROJECT SCOPE	Niall Spartan	Responsible	Accepted	
Establish the PROJECT SCOPE	Jane Ribbington	Informed	Accepted	
Establish the PROJECT SCOPE	Erelyne Flynne	Informed	Accepted	
Establish the PROJECT SCOPE	Spartacus Jundry	Informed	Accepted	
Establish the PROJECT SCOPE	Needlington Hayworth	Informed	Accepted	
Establish the PROJECT SCOPE	Simon Tarnish	Informed	Accepted	
Establish the PROJECT SCOPE	Marvyn Cloomes	Informed	Accepted	
Establish the PROJECT SCOPE	Silov Gatestata	Informed	Accepted	
Report to the Exec Board	Niall Spartan	Accountable	Communicated	

RACI DASHBOARD | 0) HELP PAGE ! | 1) RACI LOG - Fill This Out! | 2) CHECK ALLOCATION | 3) CHECK PEOPLE | LISTS

[https://business-docs.co.uk/downloads/raci-template-managing-project-responsibilities-excel/]

Tampereen yliopisto
Tampere University

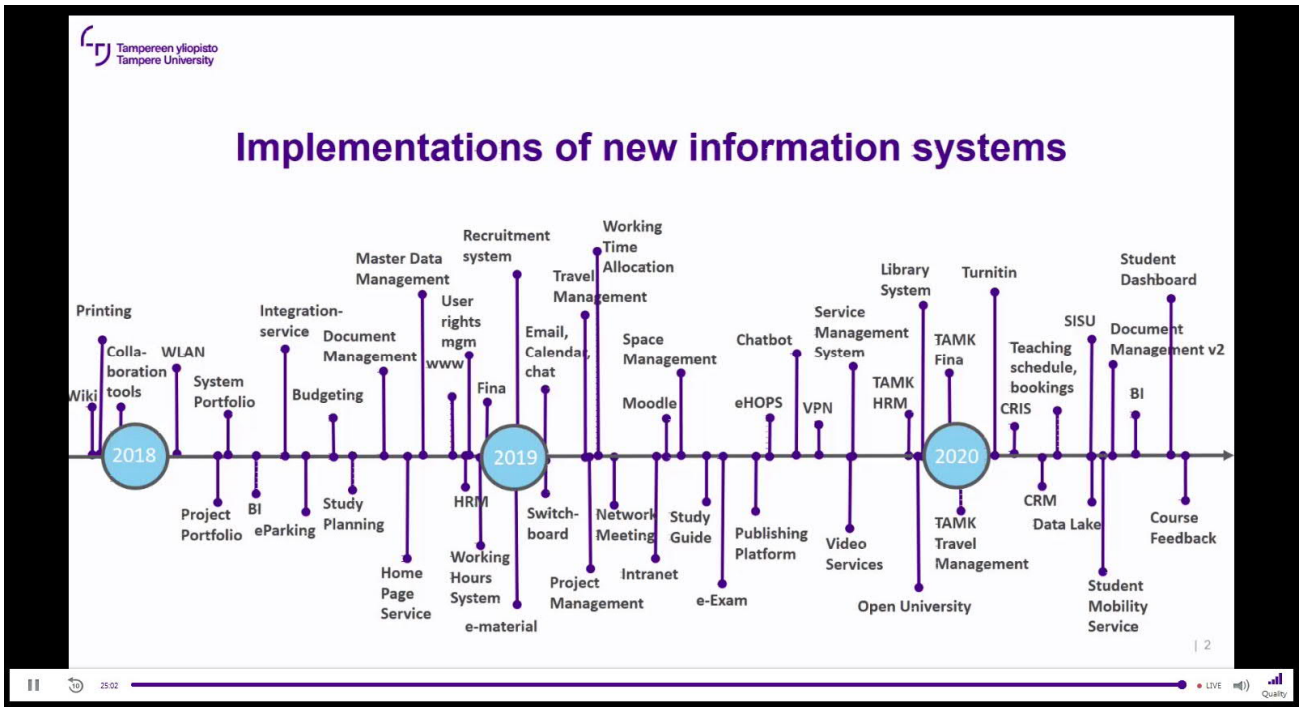
Start meeting, ”kick-off meeting”

Where to get external help ?

By the way, what if you are a project manager in a sw dev project, and the project is lacking of vital skills ? (BTW, why lacking those skills ??)

- outsourcing may not be a valid option, if you have not done it already
- starting joint research project with university is good, but slow option
- are you networked with some other companies, can you ”borrow” people
- employ known skilled people from other company (”steal” ?)
- hire a consultant (500..1000 EUR / day ?)
- get some smart fresh graduated student(s) from university (with some skills)
- employ senior researchers from university (with good skills)
- try to hire from ”free market”, social media advertising (that is public)
- buy training course for your own staff (500..1000 EUR / person / day)
- order your own staff to self-study new skills (and, at their own time ?).

Big systems should be taken into use in small steps



Factors influencing project management

- ✧ Company size
- ✧ Software customers
- ✧ Software size
- ✧ Software type
- ✧ Organizational culture
- ✧ Software development processes
- ✧ These factors mean that project managers in different organizations may work in quite different ways.

Tampereen yliopisto
Tampere University

By the way, are project workers human beings, or just ”resources” ?

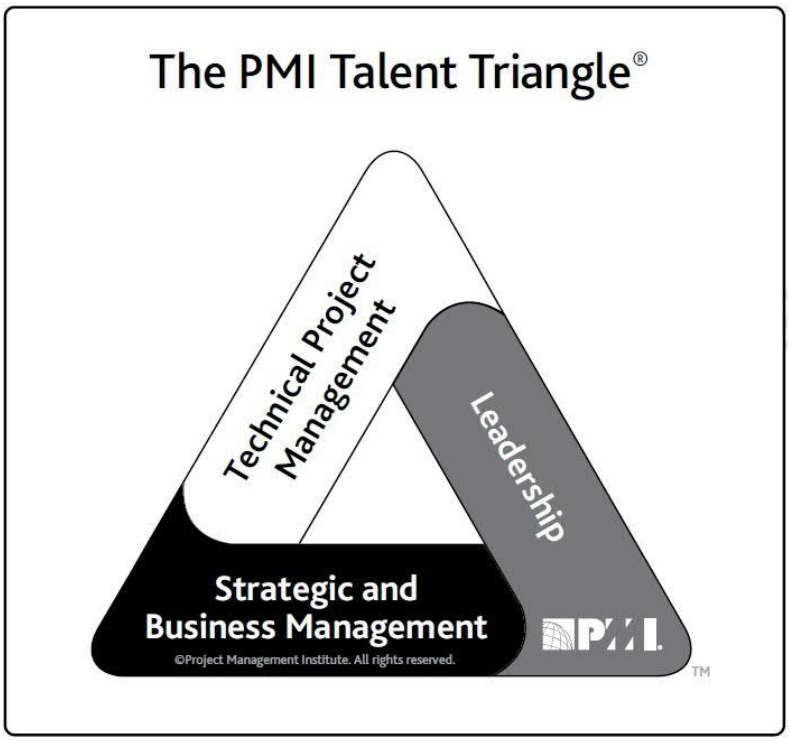


Figure 3-2. The PMI Talent Triangle®

PMI = Project Management Institute

”USA”

Tampereen yliopisto
Tampere University

Management vs. leadership

Table 3-1. Team Management and Team Leadership Compared

Management	Leadership
Direct using positional power	Guide, influence, and collaborate using relational power
Maintain	Develop
Administrate	Innovate
Focus on systems and structure	Focus on relationships with people
Rely on control	Inspire trust
Focus on near-term goals	Focus on long-range vision
Ask how and when	Ask what and why
Focus on bottom line	Focus on the horizon
Accept status quo	Challenge status quo
Do things right	Do the right things
Focus on operational issues and problem solving	Focus on vision, alignment, motivation, and inspiration

[Guide to PMBOK, 2017]

Tampereen yliopisto
Tampere University

project documentation checklist

Don't be scared, this is for heavy/big projects.

But it is a good checklist that you forget nothing.

Table 4-1. Project Management Plan and Project Documents

Project Management Plan	Project Documents	
1. Scope management plan	1. Activity attributes	19. Quality control measurements
2. Requirements management plan	2. Activity list	20. Quality metrics
3. Schedule management plan	3. Assumption log	21. Quality report
4. Cost management plan	4. Basis of estimates	22. Requirements documentation
5. Quality management plan	5. Change log	23. Requirements traceability matrix
6. Resource management plan	6. Cost estimates	24. Resource breakdown structure
7. Communications management plan	7. Cost forecasts	25. Resource calendars
8. Risk management plan	8. Duration estimates	26. Resource requirements
9. Procurement management plan	9. Issue log	27. Risk register
10. Stakeholder engagement plan	10. Lessons learned register	28. Risk report
11. Change management plan	11. Milestone list	29. Schedule data
12. Configuration management plan	12. Physical resource assignments	30. Schedule forecasts
13. Scope baseline	13. Project calendars	31. Stakeholder register
14. Schedule baseline	14. Project communications	32. Team charter
15. Cost baseline	15. Project schedule	33. Test and evaluation documents
16. Performance measurement baseline	16. Project schedule network diagram	
17. Project life cycle description	17. Project scope statement	
18. Development approach	18. Project team assignments	

[Guide to PMBOK, 2017]

Tampereen yliopisto
Tampere University

common problems

Tampereen yliopisto
Tampere University

Some kind of controlled process is needed...

Tampereen yliopisto
Tampere University

"The Standish Group report 83.9% of IT projects partially or completely fail" [www.opendoorerp.com, 2019]

All of the top factors found in failed projects include:

- Incomplete Requirements
- Lack of user involvement
- Lack of resources
- Unrealistic expectations
- Lack of executive support
- Changing Requirements & Specifications
- Lack of planning
- Didn't need it any longer
- Lack of IT management
- Technical illiteracy.

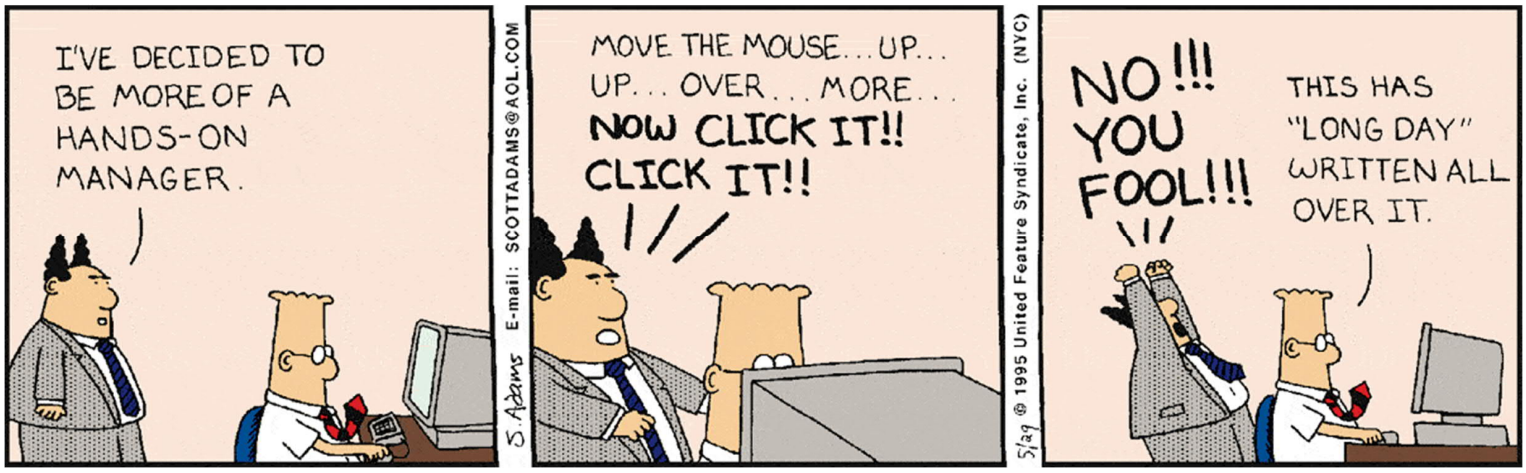
Remember:
how you define "failed"... ?

Big/long projects surely are difficult to manage.

Q: How you eat an elephant ?
A: In small pieces.

Tampereen yliopisto
Tampere University

Beware of "micromanagement"



Collected sw project data from USA

In the United States, we spend more than $250 billion each year on IT application development of approximately 175,000 projects.

The average cost of a development project

for a large company is $2,322,000;

for a medium company, it is $1,331,000; and

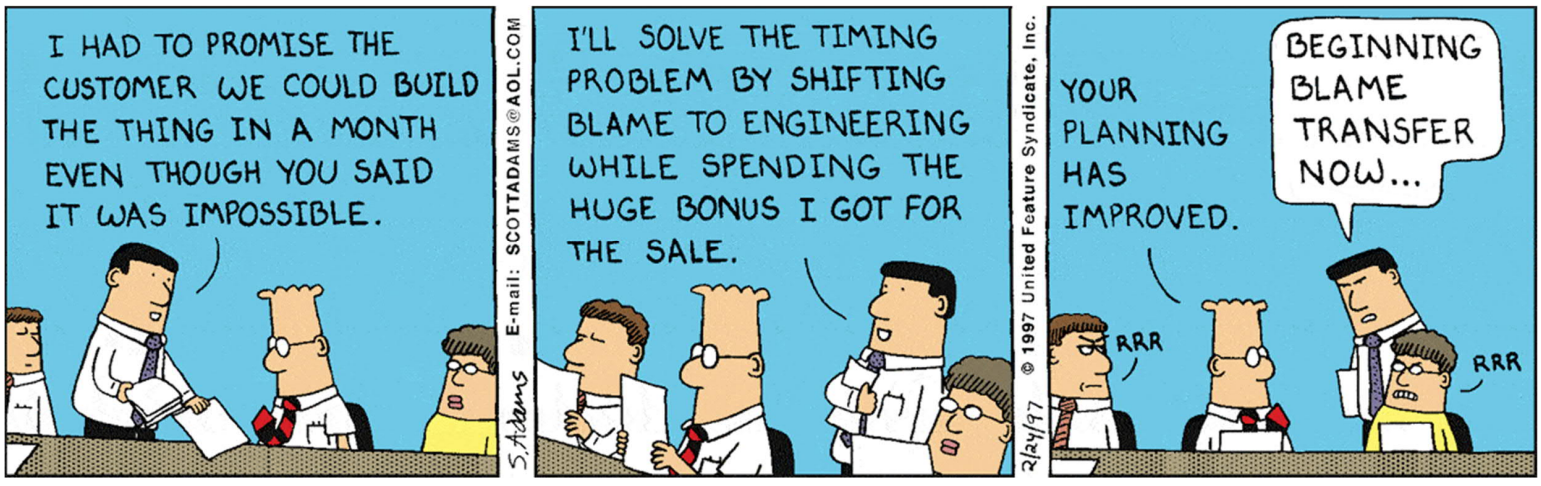
for a small company, it is $434,000.

A great many of these projects will fail. Software development projects are in chaos, and we can no longer imitate the three monkeys -- hear no failures, see no failures, speak no failures.!

The Standish Group research shows a staggering 31.1% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates.

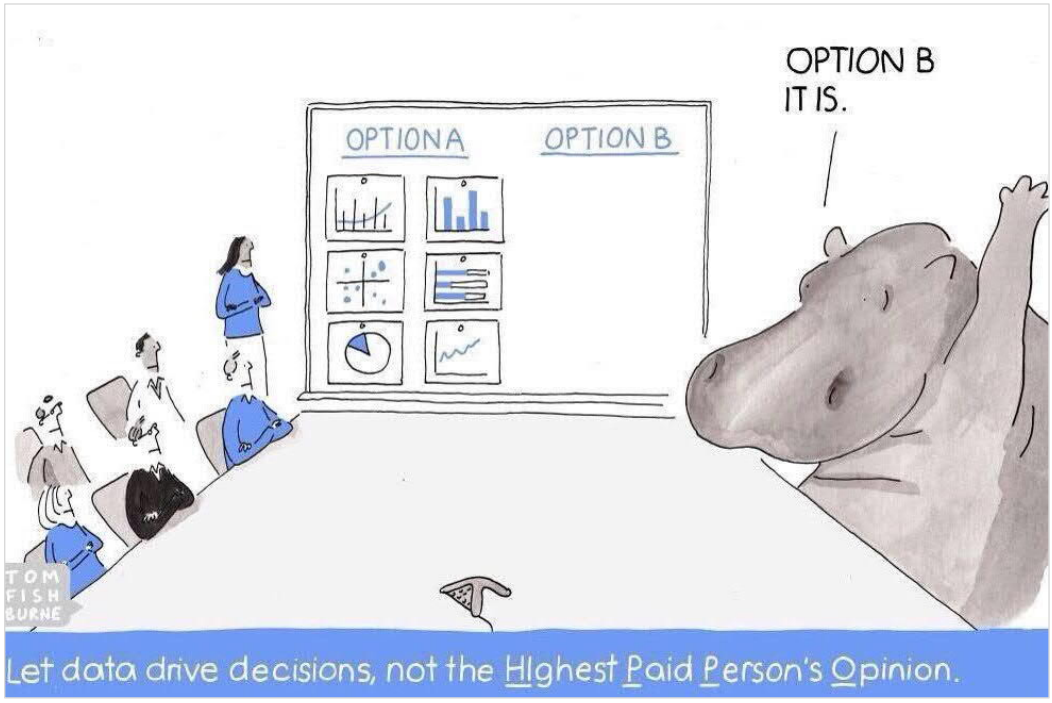
One just has to look to the City of Denver to realise the extent of this problem. The failure to produce reliable software to handle baggage at the new Denver airport is costing the city $1.1 million per day.!

[Standish Group 2014]



An old joke is that salespersons sell ”impossible” software projects, gather their bonus and carry on, or change company.

HIPPO opinion



Poor leaders or workers search for the guilty person (scapegoat).
Instead of that, you shoud put your effort on making sure such error does not happen again.

Something like this happened in old times 1990s...

Jets can now cross Pacific, Far East safe for democracy again

- By Lewis Page , Wednesday 28th February 2007 10:59 GMT

Significant new capabilities have been added to the US Air Force's F-22 "Raptor". The USAF's Raptors cost more than $300m each, and are generally thought to be the most advanced combat jets in service worldwide. However, until recently they were unable to cross the international date line (IDL) owing to a software bug in their navigation systems.

A group of F-22s heading across the Pacific for exercises in Japan earlier this month suffered simultaneous total nav-console crashes as their longitude shifted from 180 degrees West to 180 East.

Maj. Gen. Don Sheppard (ret.): "...At the international date line, whoops, all systems dumped and when I say all systems, I mean all systems, their navigation, part of their communications, their fuel systems.

It certainly could have been real serious if the weather had been bad. It turned out OK. It was fixed in 48 hours. It was a computer glitch in the millions of lines of code, somebody made an error in a couple lines of the code and everything goes.

... so that was not a big deal, just a few lines need to corrected.

Tampereen yliopisto
Tampere University

https://sourcemaking.com/antipatterns/software-project-management-antipatterns

https://www.catalyte.io/7-project-management-anti-patterns-how-to-avoid-them/

https://dzone.com/articles/5-common-antipatterns-software

https://medium.com/@l.r.s.m.2013/four-anti-patterns-that-we-need-to-avoid-working-in-software-projects-f73d56540e4e

Tampereen yliopisto
Tampere University

common success factors

Tampereen yliopisto
Tampere University

Successful project = ?

How you define a ”successful project” ? (FI: millainen on onnistunut projekti)

- in time
- within budget
- all required features done

The classic ones... not all happening often. Why ?

- customer is happy, well at least not angry, will do more business later
- project workers are happy
- project manager is happy
- real value to customer (at least some value, it is the feeling)
- project workers have learned... surely at least something
- project workers are proud of their work !
- or something else... a lot of more details may describe a successful project.

Think about that; successful project vs. successful product.

Tampereen yliopisto
Tampere University

Successful project = ???

There is no one right answer for what is a ”successful” ICT project.

Every stakeholder on project has his/her own definition for ”success”.

Remember that when e.g. reading research statistics.

If there is not much what to measure with exact values, you may consider the current project to previous ones, or ”company average”.

Anyway, in every project you do LEARN something.

Tampereen yliopisto
Tampere University

Successful project...

One important point is that from every project some kind of Final Report should be written, even a short (one page) one. It has ”lessons learnt”; what went well (and why), what did not work (and why), any good tools, techniques or methods. So new project managers can read at least what were the ”best practices” on previous projects. However, all projects are not similar, and all old rules may not apply to the next project. Successful ”best practices” may be formed to ”patterns” in company.

”Coffee table discussions” have tried to solve the old problem:

We know well what are the risks and failure reasons on software development projects. Why so many software projects still fail ? Are human beings stupid or ignorant ??

The only reason we have found, is: there are more and more new software projects established with new project managers, than current project managers get experience. Because experience is good help in projects.

Tampereen yliopisto
Tampere University

Good team =

Factors of Success/Value	Points	Investment
Small Agile Projects	25	25%
Executive Sponsorship	15	20%
Emotional Maturity	15	20%
Talented Staff	10	15%
User Involvement	9	4%
Optimization	8	4%
SAME (Standard Architectural Management Environment)	6	3%
Modest Execution	5	3%
PM/Process Expertise	4	3%
Clear Business Objectives	3	3%
Total Points & Yearly Investment	100	100%

[Standish Group, 2016]

Tampereen yliopisto
Tampere University

Project Success Factors	% of Responses
1. User Involvement	15.9%
2. Executive Management Support	13.9%
3. Clear Statement of Requirements	13.0%
4. Proper Planning	9.6%
5. Realistic Expectations	8.2%
6. Smaller Project Milestones	7.7%
7. Competent Staff	7.2%
8. Ownership	5.3%
9. Clear Vision & Objectives	2.9%
10. Hard-Working, Focused Staff	2.4%
Other	13.9%

[CHAOS report, 2015]

The definitions for these factors are:

Executive Support: when an executive or group of executives agrees to provide both financial and emotional backing. The executive or executives will encourage and assist in the successful completion of the project.

Emotional maturity is the collection of basic behaviors of how people work together. In any group, organization, or company it is both the sum of their skills and the weakest link that determine the level of emotional maturity.

User Involvement: takes place when users are involved in the project decision-making and information-gathering process. This also includes user feedback, requirements review, basic research, prototyping, and other consensus-building tools.

Optimization is a structured means of improving business effectiveness and optimizing a collection of many small projects or major requirements. Optimization starts with managing scope based on relative business value.

Skilled staff are people who understand both the business and the technology. A skilled staff is highly proficient in the execution of the project's requirements and deliver of the project or product.

CHAOS FACTORS OF SUCCESS

FACTORS OF SUCCESS	POINTS	INVESTMENT
Executive Sponsorship	15	15%
Emotional Maturity	15	15%
User Involvement	15	15%
Optimization	15	15%
Skilled Resources	10	10%
Standard Architecture	8	8%
Agile Process	7	7%
Modest Execution	6	6%
Project Management Expertise	5	5%
Clear Business Objectives	4	4%

[Standish Group, 2015]

Standish group [via www.opendoorerp.com]

The top five factors found in successful projects are:

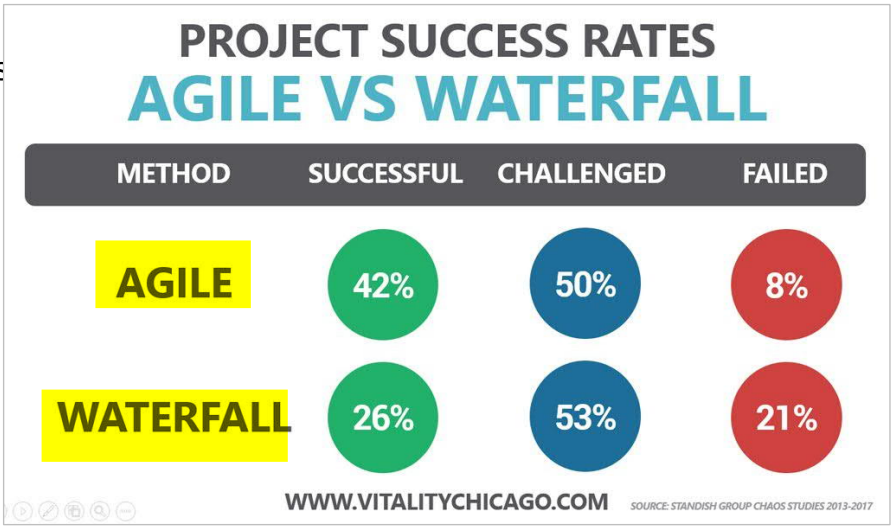
User involvement

Executive management support

Clear Statement of Requirements

Proper planning

Realistic expectations.



There is still much room for improvements; less than half of agile projects were "successful".

And once again, what does "successful" mean... ?

Tampereen yliopisto
Tampere University

What is the Standish Group definition of agile project success and failure?

Initially, the Standish Group definition of project success was limited to the triple constraint, which has been the standard for the Project Management Institute for a number of years. Using the triple constraint, the Standish Group evaluated projects as successful, challenged or failed.

Successful – A successful project was one that met all three of the triple constraints: schedule, cost, and scope.

Challenged – A challenged project would have met two out of three constraints, for example, delivered on time and on budget but not with the desired scope.

Failed – A failed project is one that is canceled before it is completed, or completed but not used.

Tampereen yliopisto
Tampere University

Success factors were:

1. Good User Participation/ Involvement.
2. Good Requirements and Specifications.
3. Good of Skills.
4. Complete Requirements.
5. Good Management and Performance from Vendor/ Contractor
6. Good of Project Planning.
7. Realistic Expectations.
8. Good of Resources.
9. Good of Executive Support (Top Management).

Table 3. Frequency Of Failure Factors Selected By Respondents

Failure factors	Frequency	Percentage	Ranking
Lack of User Involvement	38	79.17	1
Changing Requirements and Specifications	36	75.00	2
Lack of Skills	35	72.92	3
Incomplete Requirements	34	70.32	4
Vendor/ Contractor Performance and Management	32	66.67	5
Lack of Planning	30	62.50	6
Unrealistic Expectations	29	60.42	7
Lack of Resources	27	56.25	8
Lack of Executive Support (Top Management	24	50.00	9
Project Team Turn Over	18	37.50	10
Lack of Communication among project team	18	37.50	11
Project Complexity	18	37.50	12
Lack of User Knowledge	17	35.42	13
Lack of IT Management	16	33.33	14
Technology Illiteracy	15	31.25	15
Inter Department Communication	15	31.25	16

[Shamsudin Md Sarif et. Al.: Investigation of Success and Failure Factors in IT Project Management, 2018]

project management skills

By the way, are project workers human beings, or just ”resources” ?

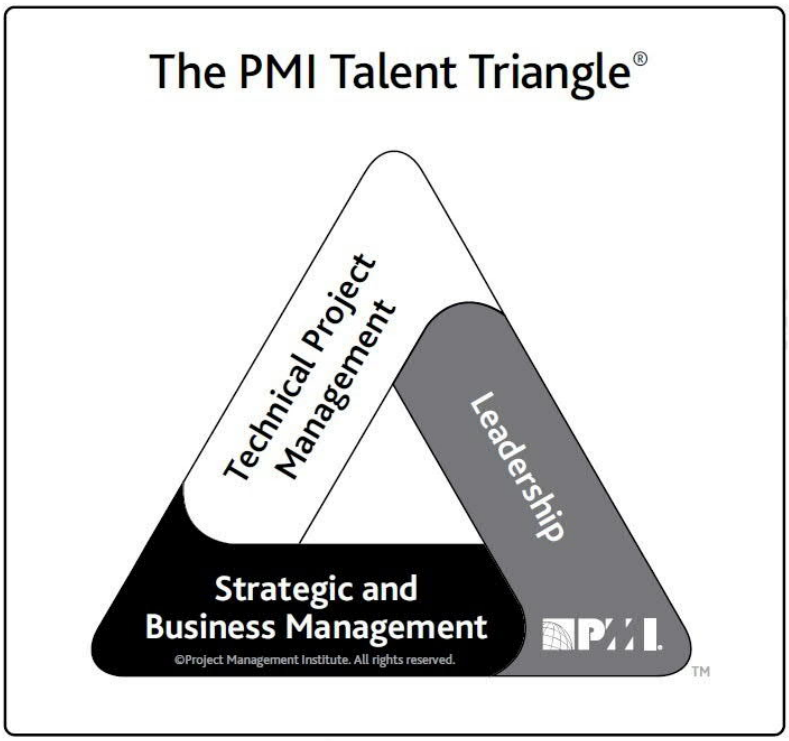


Figure 3-2. The PMI Talent Triangle®

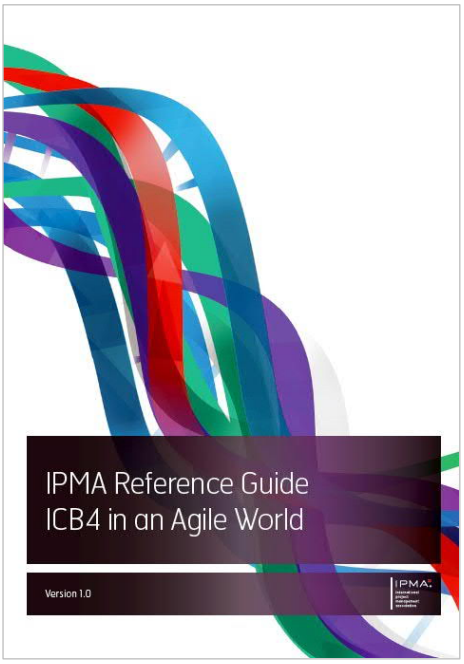
PMI = Project Management Institute

”USA”

IPMA ICB4, individual competence baseline

IPMA = International Project Management Association

”Europe”



		Knowledge	Comprehension	Application	Analysis	Synthesis	Evaluation
Perspective	Strategy						
	Governance, Structures and Processes						
	Compliance, Standards and Regulations						
	Power and Interest						
	Culture and Values						
People	Self-reflection and Self-management						
	Personal Integrity and Reliability						
	Personal Communication						
	Relations and Engagement						
	Leadership						
	Teamwork						
	Conflict and Crisis						
	Resourcefulness						
	Negotiation						
	Results orientation						
Practice	Design						
	Goals and requirements						
	Scope						
	Time						
	Organisation and Information						
	Quality						
	Finance						
	Resources						
	Procurement						
	Plan and Control						
	Risk and Opportunity						
	Stakeholders						
	Change and Transformation						

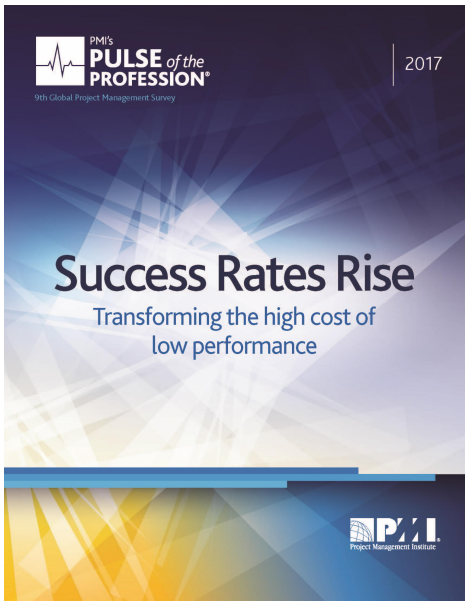
IPMA Reference guide ICB4 in an Agile World | 41

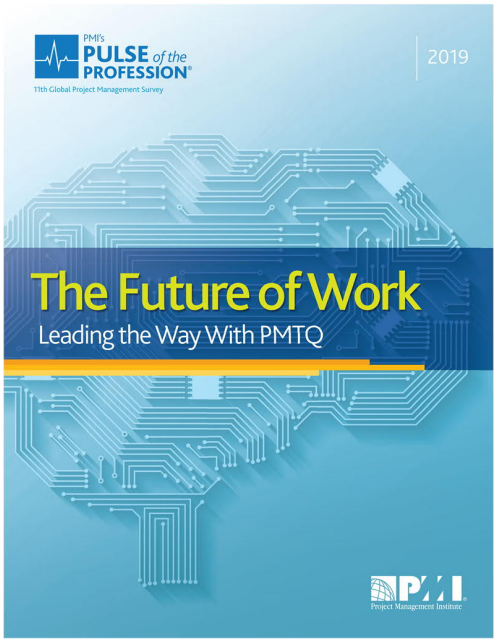
Tampereen yliopisto
Tampere University

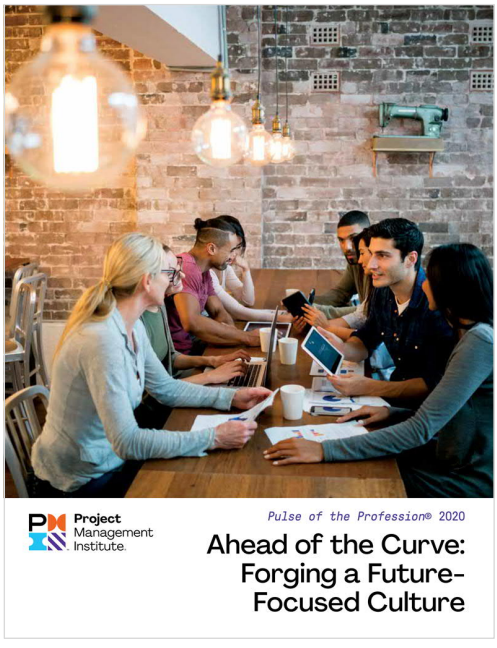
Team facilitation tips

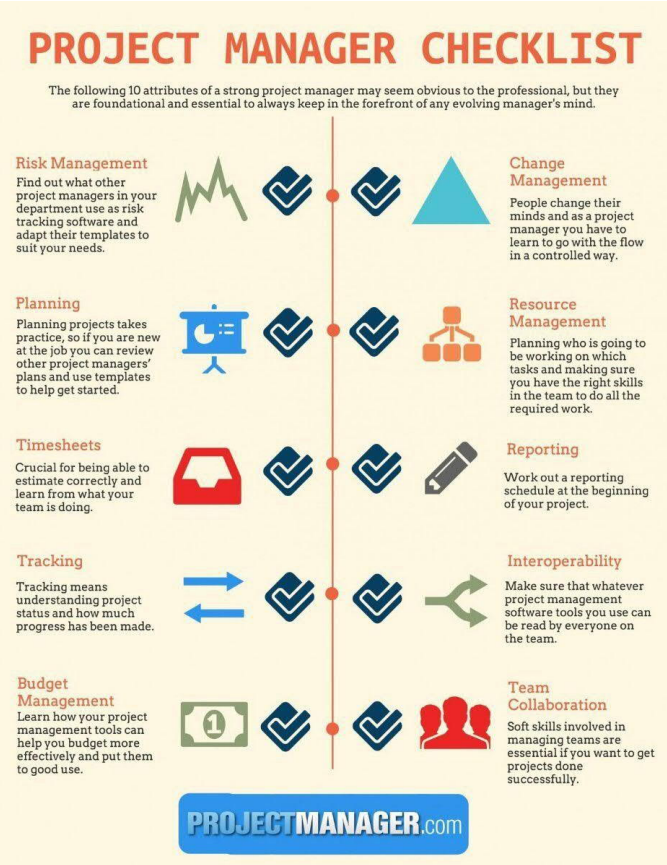
1. The team is more important than the project.
2. Have a team meeting early- within the first 3 weeks
3. Commit to meeting with teams on a regular basis
4. Look for issues and do not ignore them.
5. Recognize the positives. Use those as examples and applaud the results.
6. Guide teams through difficult conversations. Your presence is calming and safe.
7. Remain unbiased- when issues arise, everyone has some level of fault.

[www.projectmanager.com]









10 MANAGEMENT TIPS FOR GREAT LEADERS

Say thanks

People want to feel **appreciated**! A simple thank you note **doesn't cost a thing**, and it **makes a huge difference**.

Share information

Communicate the **news** that **you can**, so minds don't wander.

Raise your hand

When your people see you putting extra hours, **they are inspired** to jump in and **follow your lead**.

Remove obstacles

Bureaucracy stifles creativity and innovation. Cut down some of the paperwork.

Have fun

Your teams want to **enjoy going to work**. Play ten minutes!

Adjust your style

You have **many different communication styles and personalities** on your team. **Don't think you can manage everyone in the same way**, and don't assume everyone likes to be managed the way you like to be managed.

Empower through delegation

We know no one can do it as well as you can, BUT **you need to delegate** to give yourself time **to complete tasks** more appropriate for your level.

Set small milestones

If you can't match last year's numbers, set **milestones that can be reached**.

Give feedback

Your direct reports want feedback, and **it's crucial in making your team as productive as possible**.

Focus your time

It's the old 80:20 principle. Focus the majority of your time and attention on the **20% of your people** and projects that generate **80% of your results**.

Watch the full story about this framework on my YouTube channel © Dan Croitor 2017 • part of my video series about unique company principles and culture

VUCA for your Leadership and your strategies

What does VUCA mean for your leadership and your strategies?

VISION

Paint a picture of the future you want. Together; as a compass and for orientation; in order to confer meaning and spark motivation – and to forge internal and external identity and effectiveness

UNDERSTANDING

Understand interconnections; make them transparent. Reflect on the context. Think and plan meta-strategically. Start from the result and work backwards. Harmonize skills. Embrace and exploit behaviors and reactions. Convert anxiety and resistance to produce energy.

CLARITY

Simplicity. Focus on what counts and what it's really about. Trust, transparent connections and processes. Apply energy and force exactly where they will be most effective.

AGILITY

Flexibility. Agility. Scrutinize hierarchical management techniques. Promote a consistent culture for making decisions and accounting for mistakes. Interact transparently with objections. Facilitate innovation and build up resilience.

7 | SlideSalad.com | 2020

slidesalad

Management vs. leadership

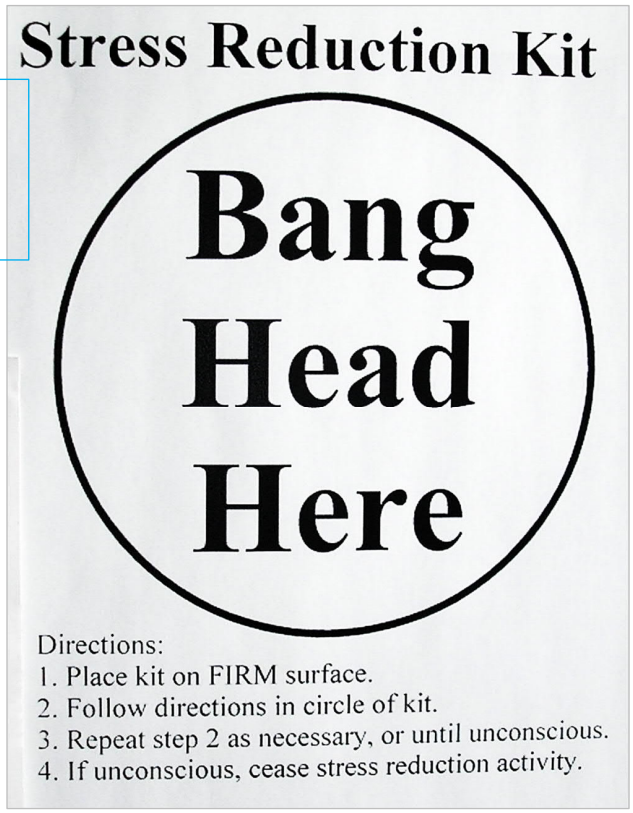
Table 3-1. Team Management and Team Leadership Compared

Management	Leadership
Direct using positional power	Guide, influence, and collaborate using relational power
Maintain	Develop
Administrate	Innovate
Focus on systems and structure	Focus on relationships with people
Rely on control	Inspire trust
Focus on near-term goals	Focus on long-range vision
Ask how and when	Ask what and why
Focus on bottom line	Focus on the horizon
Accept status quo	Challenge status quo
Do things right	Do the right things
Focus on operational issues and problem solving	Focus on vision, alignment, motivation, and inspiration

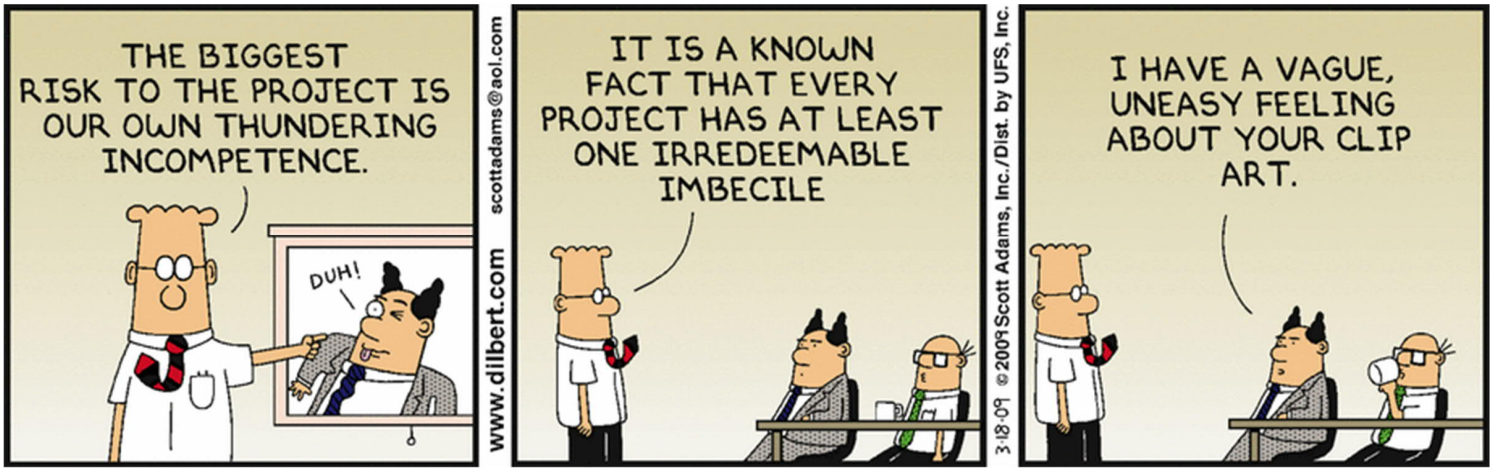
[Guide to PMBOK, 2017]

[https://borsa401k.com/]

This SRK was made mainly for project managers, but it works also for other project personnel and stakeholders.



But sometimes project termination or cancellation would be a good option.



Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

COMP.SE.100-EN (ItSE)
Introduction to Software Engneering

Lecture 10, 11.11.2020

Tensu: remember to pause Zoom lecture recording

Zoom lecture break, 10 minutes stretching, walking, etc.

Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

tools

Tampereen yliopisto
Tampere University

PM tools

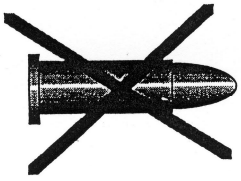
There are a lot of different project management tools available.

Mostly those are visualising tools (Gantt, WBS, BurnUp, BurnDown,…), where you should input dates and amounts of effort. So the tools do not plan the project for you.

Do not underestimate the power of ”old” spreadsheet programs, with macros.

According to experienced Project Managers, the best tools are
- coffee room table
- pen and paper
- spreadsheet.

There is no silver bullet!



No tool, no method will save you from having to think!

Tampereen yliopisto
Tampere University

Tools and technologies: .NET, Xamarin, C#, Azure, Swagger, Azure DevOps, Azure SQL, Docker, PostgreSQL
Lines of code: 6 500 (+ 40 000 autogenerated)
Classes: 70 (+ 230 autogenerated
Pull requests: 60
Feeling after project: Xamarin is bad for complex UI.

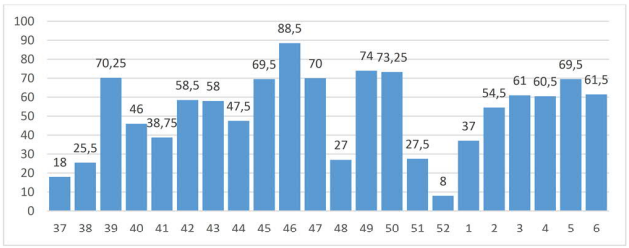


Figure 2: hours per week

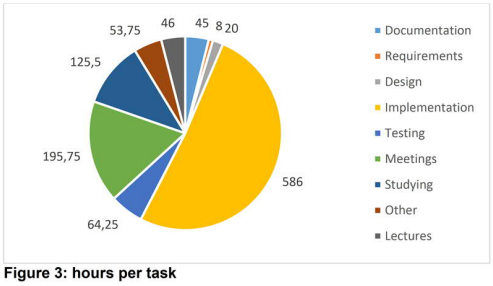
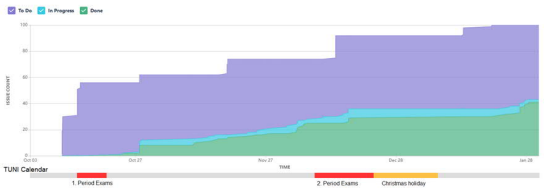


Figure 3: hours per task

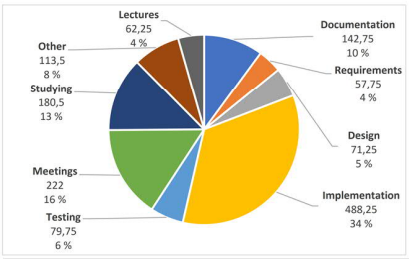
TIE-PROJ 2019-20, 1/5

Lines of code: 42183

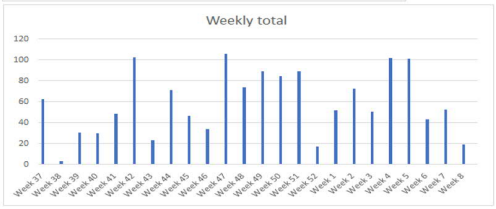


Number of jsx-component-files: 36
Number of implemented items: 51
Number of obsolete items: 20
Number of commits in version control: 188 commits
Number of revisions in version control: 588
Feelings: The whole thing ended up well

Number of methods: 153
Number of not implemented items: 29



Tools and technologies for implementation: Figma, Visual Studio Code, React, NodeJS, Redux, Express, PostgreSQL, AWS, Docker, ESLint, Prettier, Jira, Confluence, Gitlab, Microsoft Office 365, Google Drive, Telegram, JSON Web Token



Last modified: 17.02.2020 20:00

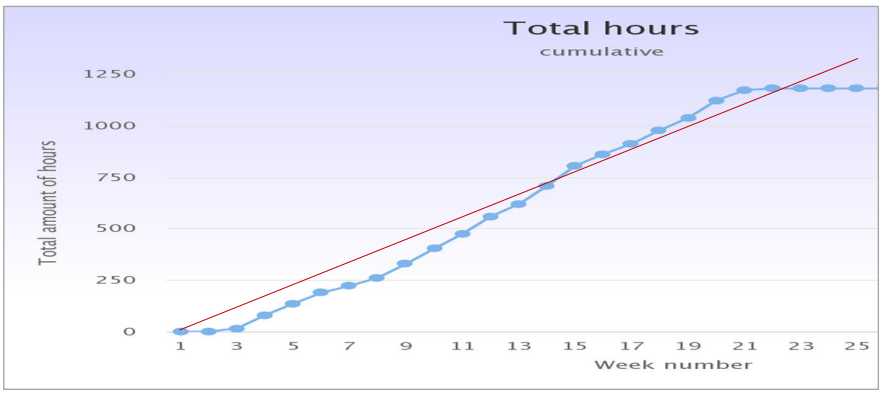
24/24

Tampereen yliopisto
Tampere University

TSS 2020 project as an example.

Weeks are real calendar weeks.

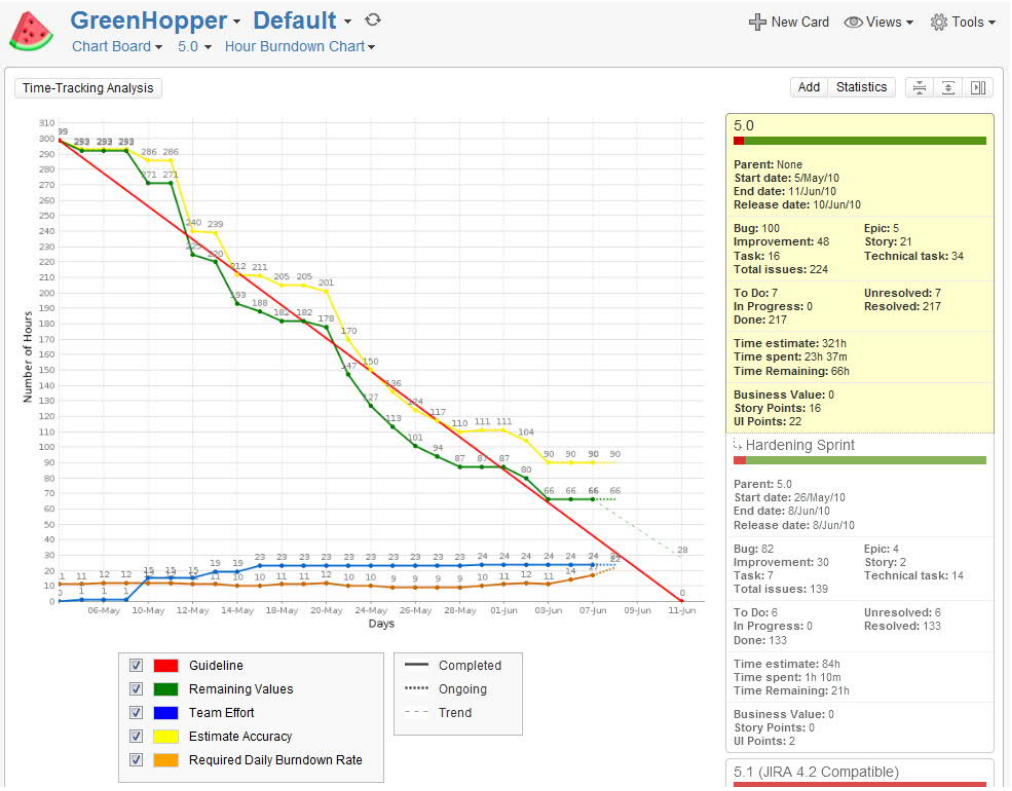
"default" amount (10 h / person)



"default" amount

Tampereen yliopisto
Tampere University

Burndown chart and working hours



Tampereen yliopisto
Tampere University

scheduling, time estimations

Tampereen yliopisto
Tampere University

scheduling, time estimations

Estimating project variables is more or less guessing (”questimation”).

If you have statistics from similar previous projects, you get some idea of the numbers (”about that much”).

You may ask experienced colleague for his/her estimation (”academic guess”).

You may gather together with colleagues and discuss about each estimated matter. You may use Planning Poker here, too.

New project managers typically do longer and detailed estimations, while experienced project managers do shorter and more general estimations (as by their working experience).

Tampereen yliopisto
Tampere University

Some effort estimation techniques

Table 7. SDEE techniques potentially suitable for small software organizations.

Techniques	Classificassion Dimensions	Licence			Data				Parametric				Expert			Model			Analogy			Regression			Composite			Artificial Intelligence		
	Techniques / Values	Proprietary	Non-Proprietary	Unknown	Many data	Sparse Data	No Data	Unknown	Parametric	Non-Parametric	Semi-Parametric	Unknown	Need Expert	No Need Expert	Unknown	Model-Based	No Model-Based	Unknown	Analogy-Based	No Analogy-Based	Unknown	Regression-Based	No Regression-Based	Unknown	Composite Technique	Single Technique	Unknown	AI-Based	No AI-Based	Unknown
	Delphi		x				x			x			x				x			x			x			x			x	
	Work Breakdown Structure (WBS)		x				x			x			x				x			x			x			x			x	
	Rule Based			x		x						x	x					x			x			x			x			x
	Guesstimation		x				x			x			x				x			x			x			x			x	
	Estimeeting		x			x				x			x				x		x				x			x			x	
	Planning Game		x				x			x			x				x			x			x			x			x	
	Expert Judgment		x				x			x			x				x			x			x			x			x	
	Analytic Hierarchy Process (AHP)		x			x				x			x				x			x			x			x			x	
	Planning Poker		x				x			x			x				x			x			x			x			x	
	Constructive Agile Estimation Algorithm		x			x			x				x			x				x		x			x				x	
	COBRA – Cost Estimation Benchmarking and Risk Analysis	x				x			x				x			x				x					x				x	
	Colaborative Filtering		x			x					x		x				x		x					x			x			x
	Use Case Points (UCP)		x			x					x		x			x			x				x		x				x	
	UCP Method Modification		x			x					x		x			x			x				x		x				x	
	Bagging + M5P/Model trees (MT)		x			x						x		x		x				x			x		x			x		
	Random + MLP		x			x						x		x		x				x			x		x			x		
	Bagging + Adaptive neuro fuzzy inference system (ANFIS)		x			x						x		x		x				x		x			x				x	
	SUMMARY	17	101	1	102	11	6	0	35	36	19	29	20	98	1	97	20	2	16	91	12	36	59	24	70	32	17	37	63	19
			119			119				119				119			119			119			119			119			119	

[Vera et.al.: Survey of Software Development Effort Estimation Taxonomies]

Measurements/estimations in Software Engineering

- You have to estimate forthcoming required effort somehow, so that you probably will meet the deadline of the project.
- Luckily in agile methods you have backlog; you do as much as you can in the sprints, and that's it.
- But to select backlog items smartly, that is art.
- If you can't measure, or don't know how to do it, you have to make "academic guesses", that is estimate.

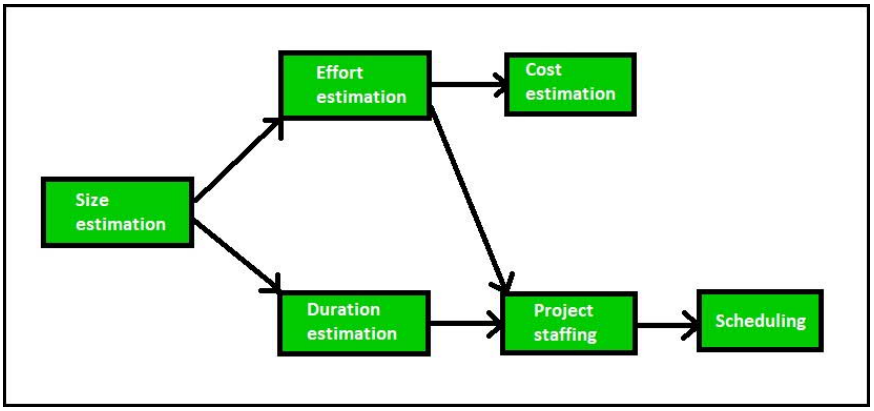
  "First estimates are more or less guessing."
- So first estimates are "questimates", but they are something; next estimates: lower or higher ?
- Experience from projects naturally is of big help, but newcomers just have to try their best.

Tampereen yliopisto
Tampere University

project monitoring and control

Project Estimation: Project Size Estimation is the most important parameter based on which all other estimations like cost, duration and effort are made.

[https://www.geeksforgeeks.org/software-engineering-role-and-responsibilities-of-a-software-project-manager/]



- **Scheduling:** After completion of estimation of all the project parameters, scheduling for manpower and other resources are done.
- **Staffing:** Team structure and staffing plans are made.
- **Risk Management:** The project manager should identify the unanticipated risks that may occur during project development risk, analysis the damage might cause these risks and take risk reduction plan to cope up with these risks.
- **Miscellaneous plans:** This includes making several other plans such as quality assurance plan, configuration management plan, etc.

Estimating software projects... 1

”Software projects are difficult to estimate.”

What would we like to know ?

- duration (calendar time)
- effort (person-months/person-hours)
  – > how many workers are needed
  – > what would it cost ?

So if we can estimate size, we know all:

- size -> effort -> calendar time -> costs.

Estimating software projects... 2

Coding work is difficult to estimate... ?

HOW did YOU estimate tasks during project assignment Sprint 1 ?

- ...?
- ..?...
- ?...

Software folklore: ”a trained monkey would do this in three months !”

Estimating software projects... 3

The best way to accurate estimations is collecting data from several previous projects, and have them as a basis for next estimations.

But if you are a newcomer, you only have

- guessing (Stetson method = in Finnish ”vedetään hatusta/hihasta”, planning poker,...)
- think hard with your colleagues, many times and from many roles/viewpoints (code-test-document, top-down, bottom-up)
- ask some guru(s), senior developers (good option)
- general software engineering metrics calculations (make several of them).

Effort estimation

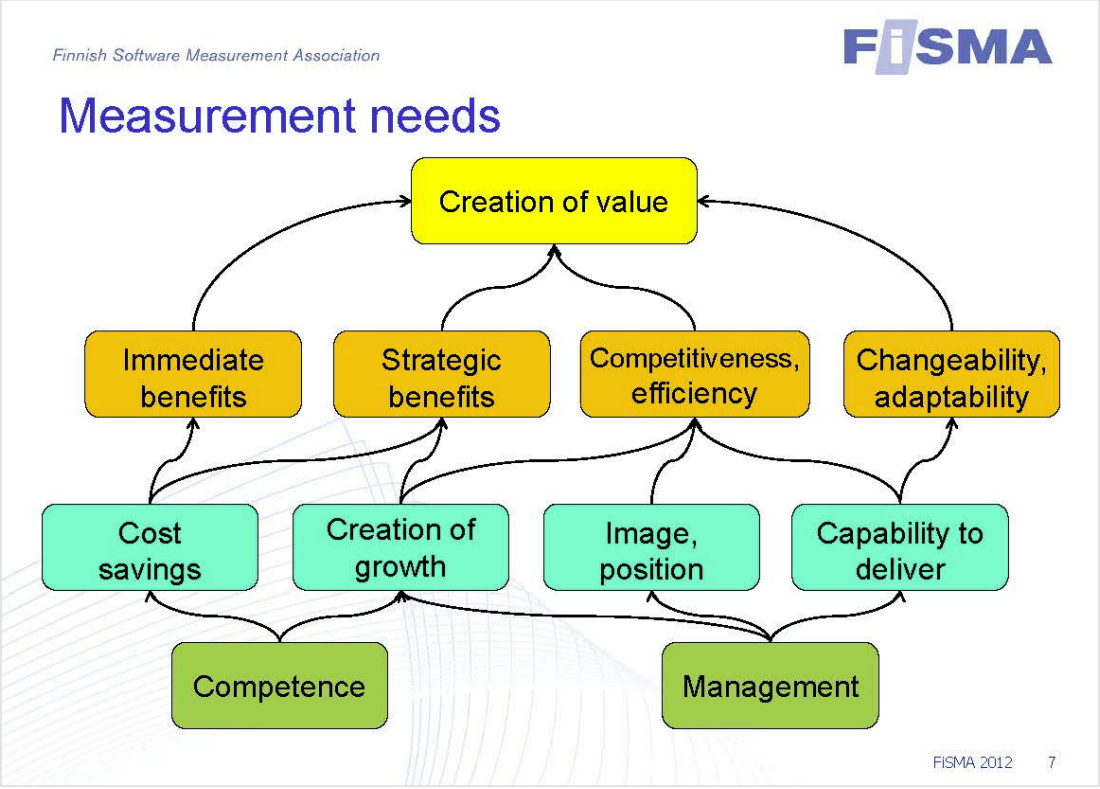
There are several well-known calculations for estimating effort in software engineering (out of tens if not hundreds);

- FPA
- FiSMA
- COCOMO.

However, if you want accurate estimations, you should calculate and add to those some organisation/team-specific variables/constants from past projects, to fine-tune the ”standard” calculations. Naturally those are valid only for ”similar type” projects.

metrics

FiSMA = Finnish Software Measurement Association



Tampereen yliopisto
Tampere University

Top 5 Software Metrics to Manage Development Projects

Types of Software Metrics

- Formal code metrics—Such as Lines of Code (LOC), code complexity, Instruction Path Length, etc. In modern development environments, these are considered less useful.
- Developer productivity metrics—Such as active days, assignment scope, efficiency and code churn. These metrics can help you understand how much time and work developers are investing in a software project.
- Agile process metrics—Such as lead time, cycle time and velocity. They measure the progress of a dev team in producing working, shipping-quality software features.
- Operational metrics—Such as Mean Time Between Failures (MTBF) and Mean Time to Recover (MTTR). This checks how software is running in production and how effective operations staff are at maintaining it.
- Test metrics—Such as code coverage, percent of automated tests, and defects in production. This measures how comprehensively a system is tested, which should be correlated with software quality.
- Customer satisfaction—Such as Net Promoter Score (NPS), Customer Effort Score (CES) and Customer Satisfaction Score (CSAT). The ultimate measurement of how customers experience the software and their interaction with the software vendor.

[https://www.sealights.io/software-development-metrics/top-5-software-metrics-to-manage-development-projects-effectively/]

Tampereen yliopisto
Tampere University

metrics

Traditionally metrics in software projects have been based on estimating required working hours; from that it was possible to estimate cost and schedule.

Some famous metrics are

- COCOMO, COCOMO II = COnstructive COst MOdel
- FPA = Function Point Analysis
- FiSMA FPA = Function Point Analysis by Finnish Software Metrics Association
- and there are much more…

Nowadays the ”best” metrics may well be Project Manager’s and Developer Team’s experience.

We all know that LOC is not a good metrics, but at least it is better than nothing.

Tampereen yliopisto
Tampere University

some general metrics

COCOMO = Constructive cost model, COCOMO II (2000)

- includes formulas for calculating required effort and thus schedule
- based on estimating system's LOC (lines of code), huge amount of project statistics
- many variables according organisation; e.g. complexity and skills
- naturally depends also from programming language.

FPA = Function Point Analysis

- system's functions are counted
- function points can be converted to LOC.

FiSMA FPA = Function Point Analysis by Finnish Software Metrics Association

Tampereen yliopisto
Tampere University

Some metrics www calculators

UofSC www calculator for COCOMO II

https://csse.usc.edu/tools/COCOMOII.php

NASA STRS www page for COCOMO calculation

https://strs.grc.nasa.gov/repository/forms/cocomo-calculation/

UofM Basic COCOMO calculators

http://groups.umd.umich.edu/cis/tinytools/cocomo.html

http://www.edtechnology.in/cocomo/

UofM Function Point calculator

http://groups.umd.umich.edu/cis/course.des/cis375/projects/fp99/main.html

Some COCOMO and FP www calculators

http://csse.usc.edu/tools/COCOMOII.php

http://csse.usc.edu/tools/COCOMOSuite.php

http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/artan/functionpoints.htm

http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/harvey/FP_Calc.html

Some Function Point www calculators

UofM Function Point calculator
http://groups.umd.umich.edu/cis/course.des/cis375/projects/fp99/main.html

UofM FP calculator
http://groups.umd.umich.edu/cis/course.des/cis525/js/f00/harvey/FP_Calc.html

JMU Function Point calculator
https://w3.cs.jmu.edu/bernstdh/web/common/webapps/oop/fpcalculator/FunctionPointCalculator.html

Function point, fp

FPA (1979-) and FiSMA (2003-) are using function points.

Function points

- measure software size based on the functionality
- are derived from requirements
- are language independent.

Example metrics are

- effort per function point
- defects per function point
- cost per function point.

www.ifpug.org

Function point (FP) (FI: toimintopiste)

A Function Point (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure software size. They are widely accepted as an industry standard for functional sizing.

For sizing software based on FP, several recognized standards and/or public specifications have come into existence.

ISO Standards (as of 2013)

- COSMIC – ISO/IEC 19761:2011 Software engineering. A functional size measurement method.
- FiSMA – ISO/IEC 29881:2008 Information technology - Software and systems engineering - FiSMA 1.1 functional size measurement method.
- IFPUG – ISO/IEC 20926:2009 Software and systems engineering - Software measurement - IFPUG functional size measurement method.
- Mark-II – ISO/IEC 20968:2002 Software engineering - Ml II Function Point Analysis - Counting Practices Manual.
- NESMA – ISO/IEC 24570:2005 Software engineering - NESMA function size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis.

[www.tutorialspoint.com/estimation_techniques/]

FiSMA (2003-)

- Originated from Finnish Software Measurement Association's models
- is one Functional Size Measurement (FSM) method
- "FiSMA 1.1 is based purely on Functional User Requirements (FUR). User requirements can be thought of as functional (what the software does) and non-functional (how the software must perform, including quality requirements)."

SUOMEN STANDARDISOIMISLIITTO SFS | STANDARDI | **SFS-ISO/IEC 29881**

Suomen Standardisoimisliitto SFS ry
Finnish Standards Association SFS

Vahvistettu
2013-01-21

1 (1 + 16)

SFS/ICS 35.080

Tämä standardi on vahvistettu englanninkielisenä | *This standard is approved in English*

INFORMATION TECHNOLOGY – SYSTEMS AND SOFTWARE ENGINEERING – FISMA 1.1 FUNCTIONAL SIZE MEASUREMENT METHOD

Tämä standardi sisältää kansainvälisen standardin ISO/IEC 29881:2010 "Information technology – Systems and software engineering – FiSMA 1.1 functional size measurement method" englanninkielisen tekstin.

Kansainvälinen standardi ISO/IEC 29881:2010 on vahvistettu suomalaiseksi kansalliseksi standardiksi.

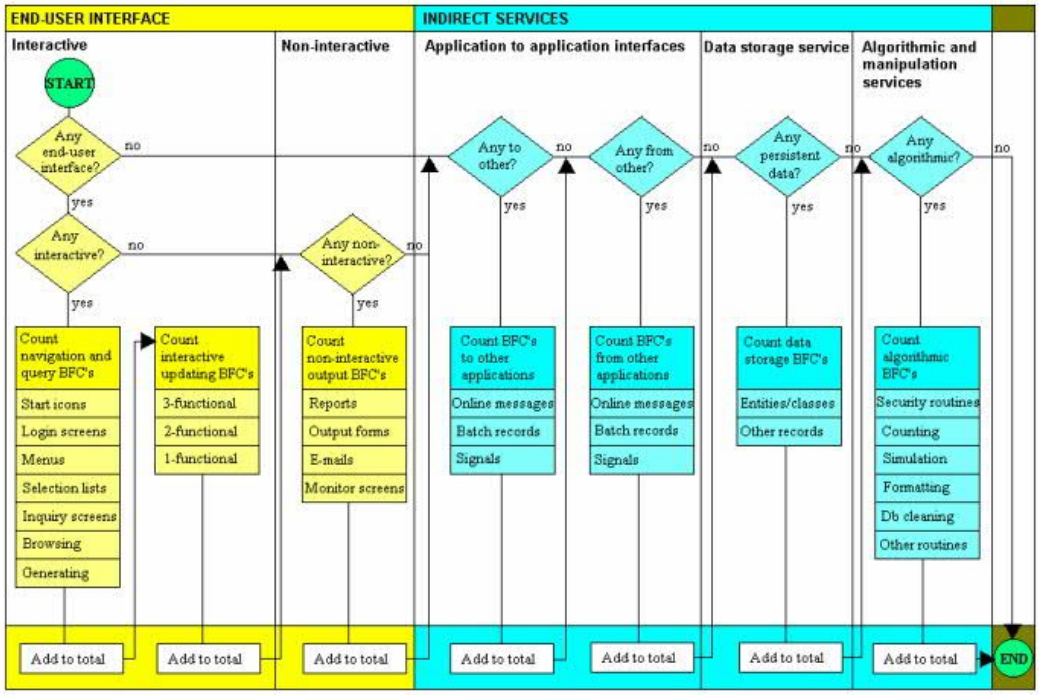
This standard consists of the English text of the International Standard ISO/IEC 29881:2010 "Information technology – Systems and software engineering – FiSMA 1.1 functional size measurement method".

The International Standard ISO/IEC 29881:2010 has the status of a Finnish national standard.

Seven parts of FiSMA

1. Interactive end-user navigation and query
2. Interactive end-user input
3. Non-interactive end-user output
4. Interface BFC's to other applications
5. Interface BFC's from other applications
6. Data storage services
7. Algorithmic and manipulation services.

BFC = Base Functional Component.



BFC = base functional component

Figure 4 — FiSMA 1.1 process

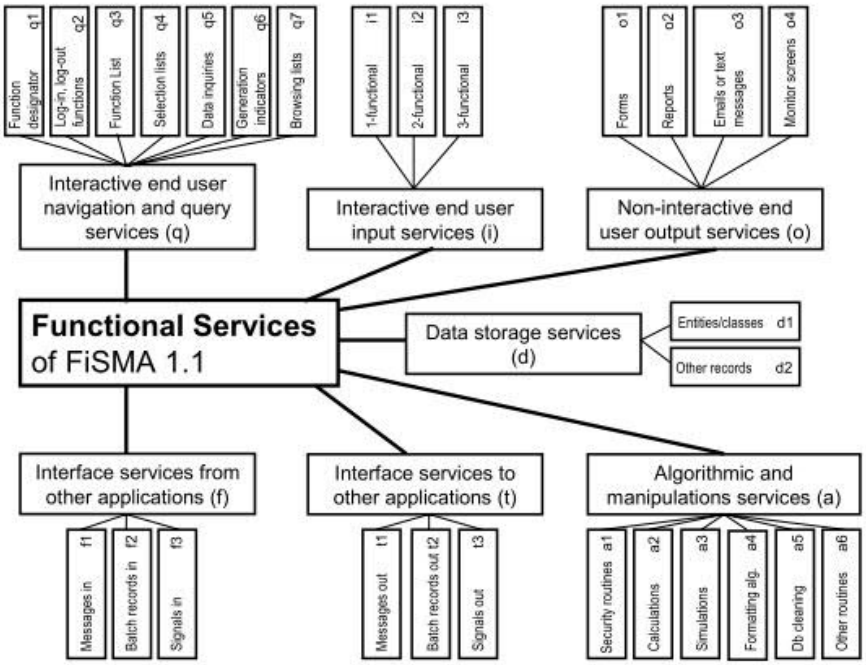
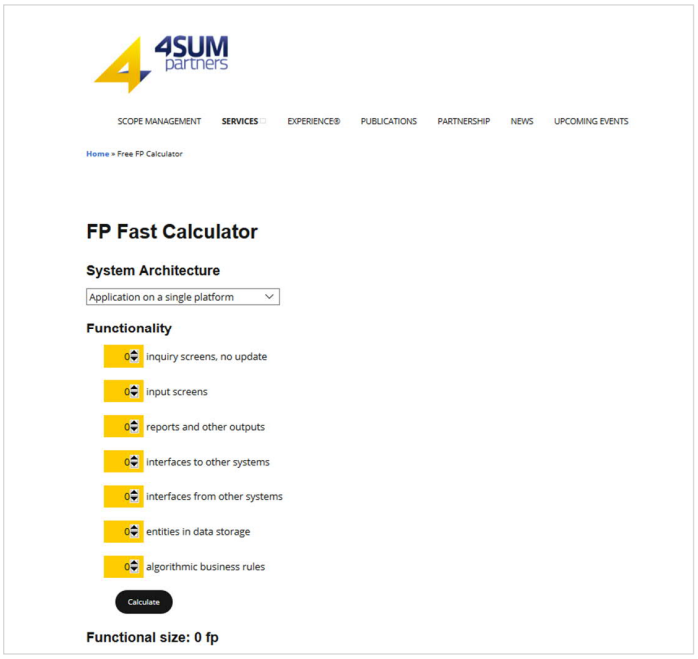
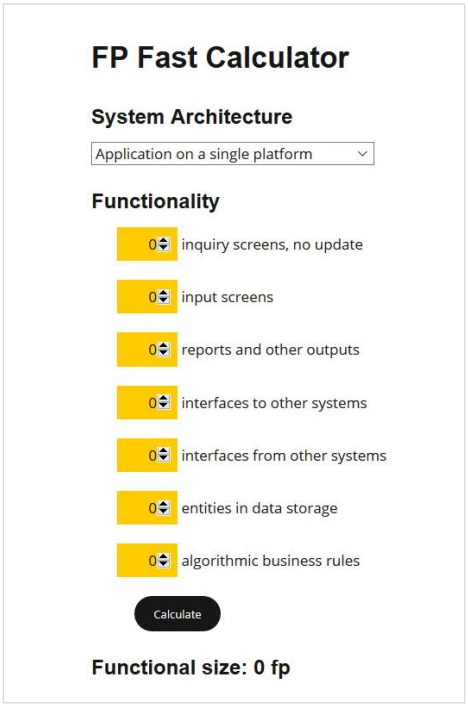


Figure 3 — FiSMA 1.1 BFC classes and BFC types

BFC = base functional component

http://www.4sumpartners.com/fpfastcalculator/





WE 7, part 4

You now have Ffp estimation/calculation from FiSMA model.

By Sw Eng folklore (statistics) one FP may be about 50 LOC Java (but beware, this is just a rough conversion for just one language).

Folklore also claims, that it takes 10 person-hours to produce one FP.

- So how many LOC would the planned system require and need (what is the size) ?

WE 7, part 4,2

More data from other sources about LOC/FP:

- C : 128 LOC / 1 FP
- C : 125 LOC / 1 FP
- C : 148 LOC / 1 FP
- C++ : 64 LOC / 1 FP
- C++ : 59 LOC / 1 FP
- C++ : 55 LOC / 1 FP
- C# : 58 LOC / 1 FP
- Java : 55 LOC / 1 FP
- Java : 53 LOC / 1 FP.

WE 7, part 4,4

More data from other sources:

Productivity

- commercial applications
  200-900 LOC /person-month
- systems programs
  150-400 LOC /person-month
- real-time embedded systems
  40-160 LOC /person-month.

WE 7, part 4,5

More data from other sources:

Productivity

Program size 1000 LOC:

- coding : 13,26 LOC / hour
- net (whole project) : 5,76 LOC / hour

Program size 10000 LOC:

- coding : 11,36 LOC / hour
- net (whole project) : 4,13 LOC / hour

Program size 100000 LOC:

- coding : 9,09 LOC / hour
- net (whole project) : 2,60 LOC / hour.

WE 7, part 4,6

More data from other sources:

Productivity

- for systems of 50 FP size
  1,3 hours/FP
- for systems of 7000 FP size
  12,1 hours/FP
- for systems of 15000 FP size
  133,6 hours/FP.

Some statistical calculations…

According to IFPUG calculations (2013);

- **Microsoft Windows 7 = 202150 fp**
- **Microsoft Windows XP = 66238 fp**
- **F15 avionics/weapons = 23109 fp**
- **Facebook = 8404 fp**
- **Microsoft Word 1431 fp**
- **Mozilla Firefox = 1342 fp.**

IFPUG = International Function Point Users Group.

WE 7, part 5

Now you have the LOC count estimation (from FiSMA).

Calculate what would be the needed person-months (person-hours) and calendar time by COCOMO (II) metrics ?

You may also use some www calculators... see e.g.

http://csse.usc.edu/tools/COCOMOII.php

http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/artan/functionpoints.htm

SIMO Light (PROJ 2012-13)

- 6 persons
- 6 months
- 3413 SLOC, 5650 LOC
- Java, Android tablet
- 14 classes
- 205 methods
- 1188 working hours.

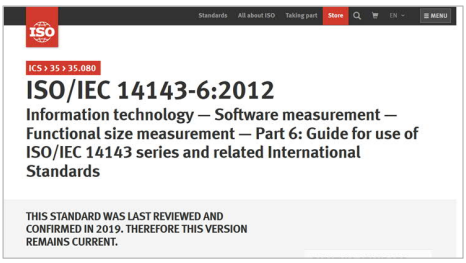
SIMO light (old WE 11, part 3)

By Sw Eng folklore one FP may be 50 LOC Java, so the SIMO Light would have 70 FP (70 by SLOC, or 110 by LOC).

Folklore also claims, that it takes 10 person-hours to produce one FP.

By that, SIMO Light would be around 700 (or 1100) working hours. We know that it took 1188 hours as a student project. Students are not yet skilled professionals, and university course has some ”course tasks” which do not exist at real sw projects. So that ”rule of thumb” is not far from reality.

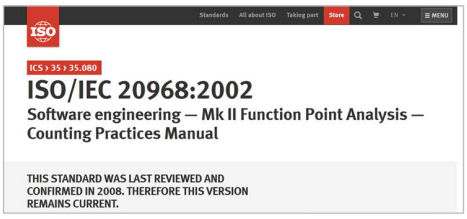
Calculate Ffp estimation by FiSMA guidelines.


ICS › 35 › 35.080

ISO/IEC 14143-6:2012

Information technology — Software measurement — Functional size measurement — Part 6: Guide for use of ISO/IEC 14143 series and related International Standards

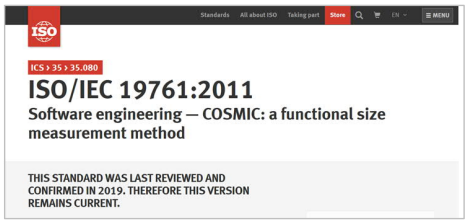
THIS STANDARD WAS LAST REVIEWED AND CONFIRMED IN 2019. THEREFORE THIS VERSION REMAINS CURRENT.


ICS › 35 › 35.080

ISO/IEC 20968:2002

Software engineering — Mk II Function Point Analysis — Counting Practices Manual

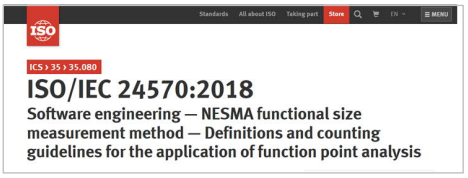
THIS STANDARD WAS LAST REVIEWED AND CONFIRMED IN 2008. THEREFORE THIS VERSION REMAINS CURRENT.


ICS › 35 › 35.080

ISO/IEC 19761:2011

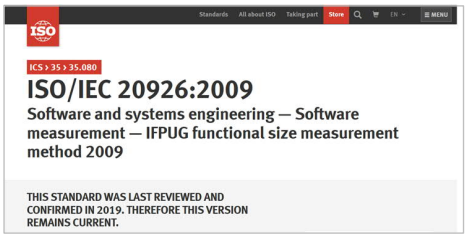
Software engineering — COSMIC: a functional size measurement method

THIS STANDARD WAS LAST REVIEWED AND CONFIRMED IN 2019. THEREFORE THIS VERSION REMAINS CURRENT.


ICS › 35 › 35.080

ISO/IEC 24570:2018

Software engineering — NESMA functional size measurement method — Definitions and counting guidelines for the application of function point analysis


ICS › 35 › 35.080

ISO/IEC 20926:2009

Software and systems engineering — Software measurement — IFPUG functional size measurement method 2009

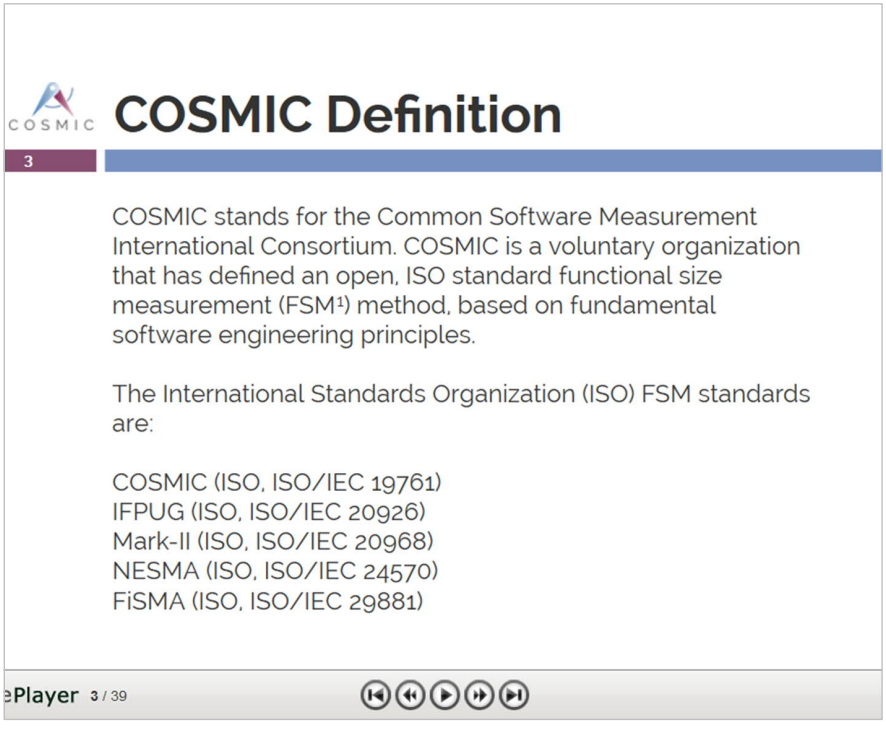
THIS STANDARD WAS LAST REVIEWED AND CONFIRMED IN 2019. THEREFORE THIS VERSION REMAINS CURRENT.


ICS › 35 › 35.080

ISO/IEC 29881:2010

Information technology — Systems and software engineering — FiSMA 1.1 functional size measurement method

THIS STANDARD WAS LAST REVIEWED AND CONFIRMED IN 2019. THEREFORE THIS VERSION REMAINS CURRENT.


COSMIC

COSMIC Definition

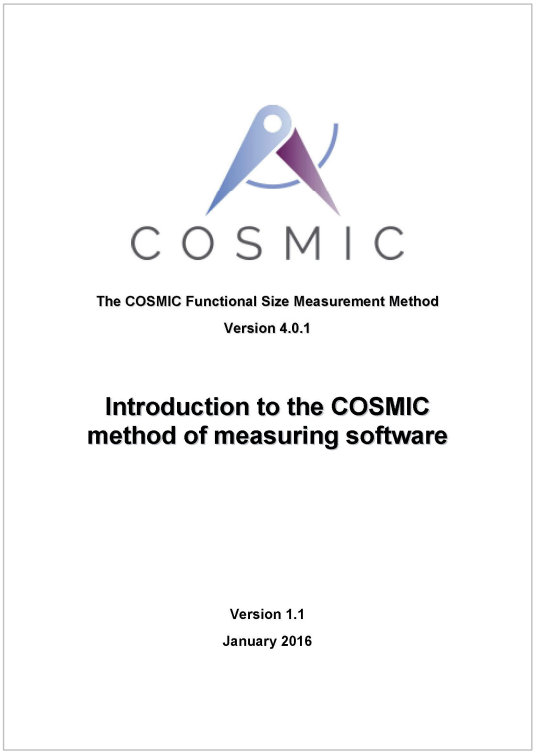
3

COSMIC stands for the Common Software Measurement International Consortium. COSMIC is a voluntary organization that has defined an open, ISO standard functional size measurement (FSM[1]) method, based on fundamental software engineering principles.

The International Standards Organization (ISO) FSM standards are:

COSMIC (ISO, ISO/IEC 19761)
IFPUG (ISO, ISO/IEC 20926)
Mark-II (ISO, ISO/IEC 20968)
NESMA (ISO, ISO/IEC 24570)
FiSMA (ISO, ISO/IEC 29881)

ePlayer 3 / 39


COSMIC

The COSMIC Functional Size Measurement Method

Version 4.0.1

Introduction to the COSMIC method of measuring software

Version 1.1

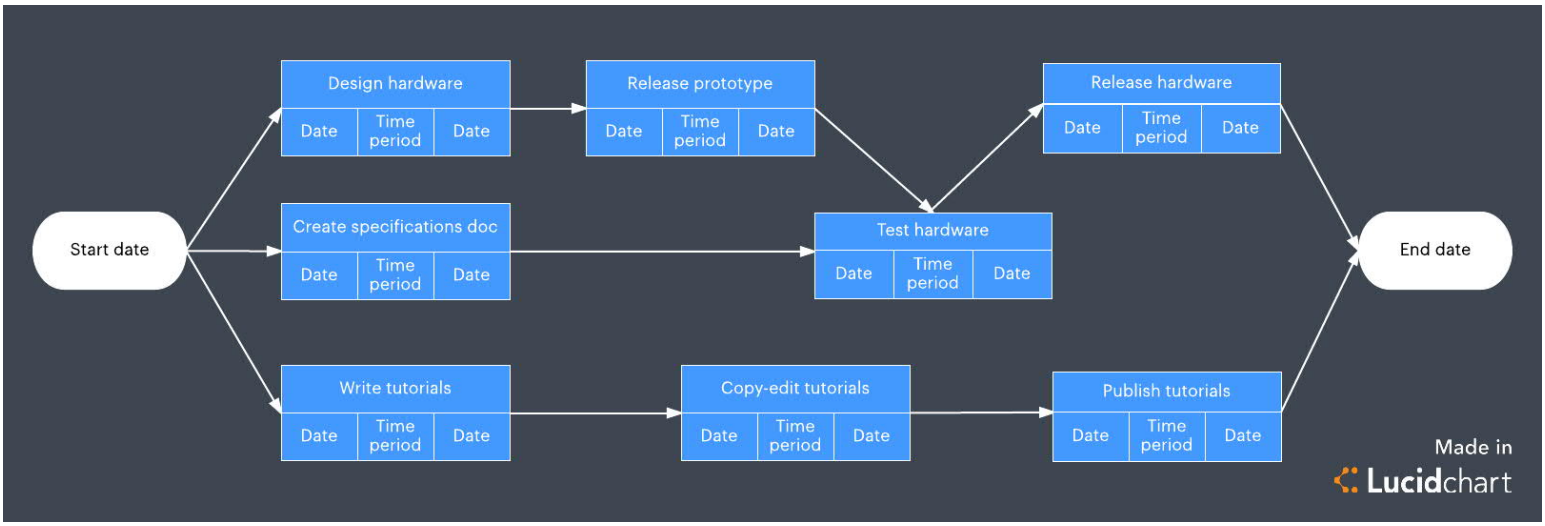
January 2016

Tampereen yliopisto
Tampere University

If you measure ”wrong” things, you do not get a glue how your project is going

Tampereen yliopisto
Tampere University

PERT (program evaluation and review technique)



Tampereen yliopisto
Tampere University

Indicators

Tampereen yliopisto
Tampere University

You may establish many kind of indicators or ”traffic lights”

How you know where your project is going ? YES ?

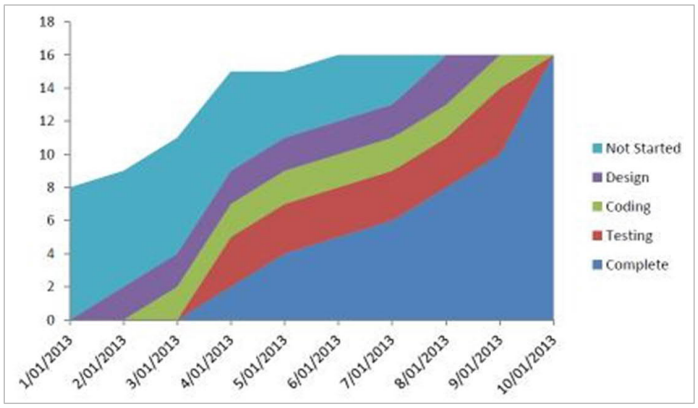
Do you care about your project only when you start hear screaming ? NO ?

Well, first you should find suitable realistic metrics what to measure during the project. For example number of requirements or features implemented and tested (= not just ”ready” but also ”done”), that is completed according to DoD.

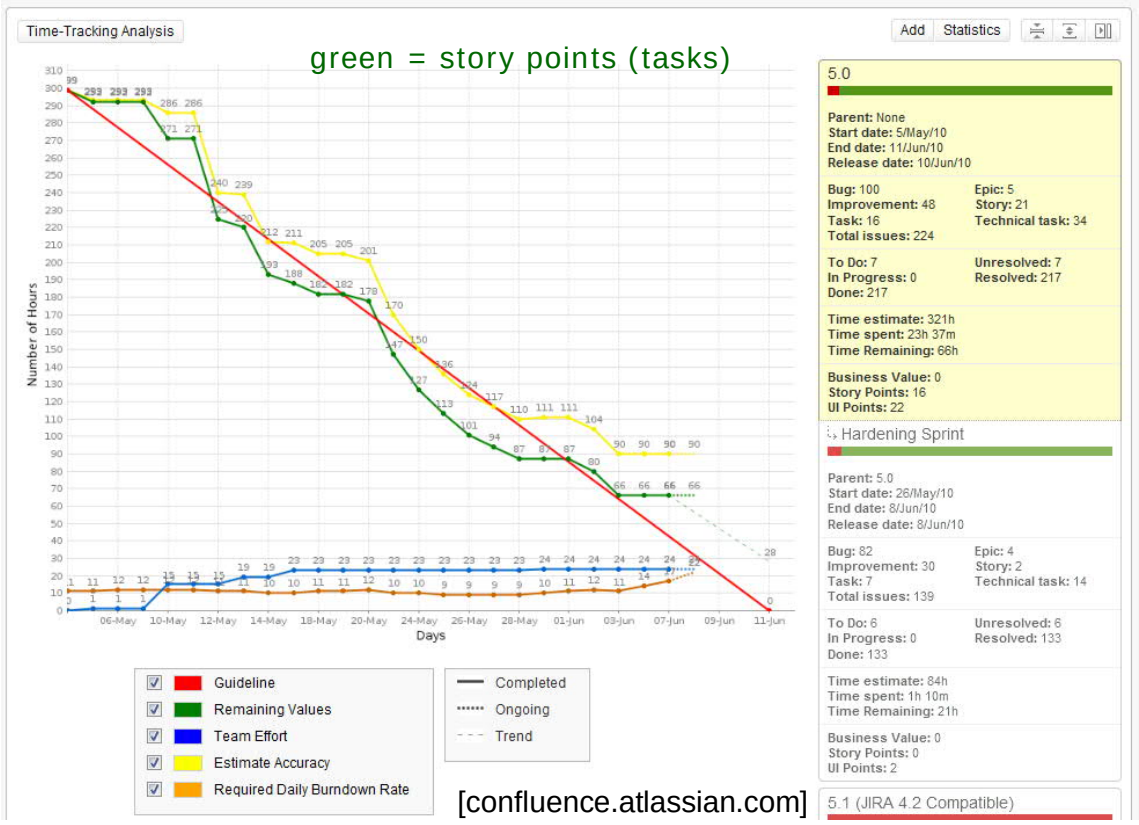
Next you may develop some automatic summaries about the numbers, which are presented in easy-to-understand and quick-to-read view (= manager proof). Something like automated CI pipelines show.

It may be enough to watch those metrics indicators weekly. One yellow is not bad, but two yellow weeks are. Go to developers and find what it means.

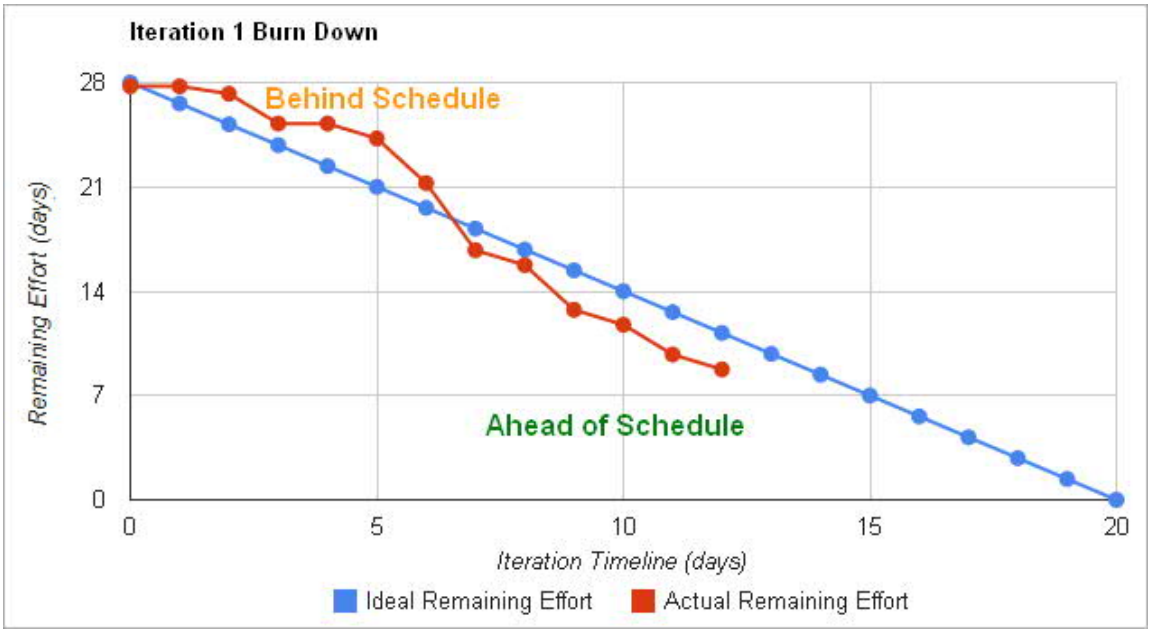
Some sw project metrics graphs



[https://www.daxx.com/blog/development-trends/top-software-development-metrics]



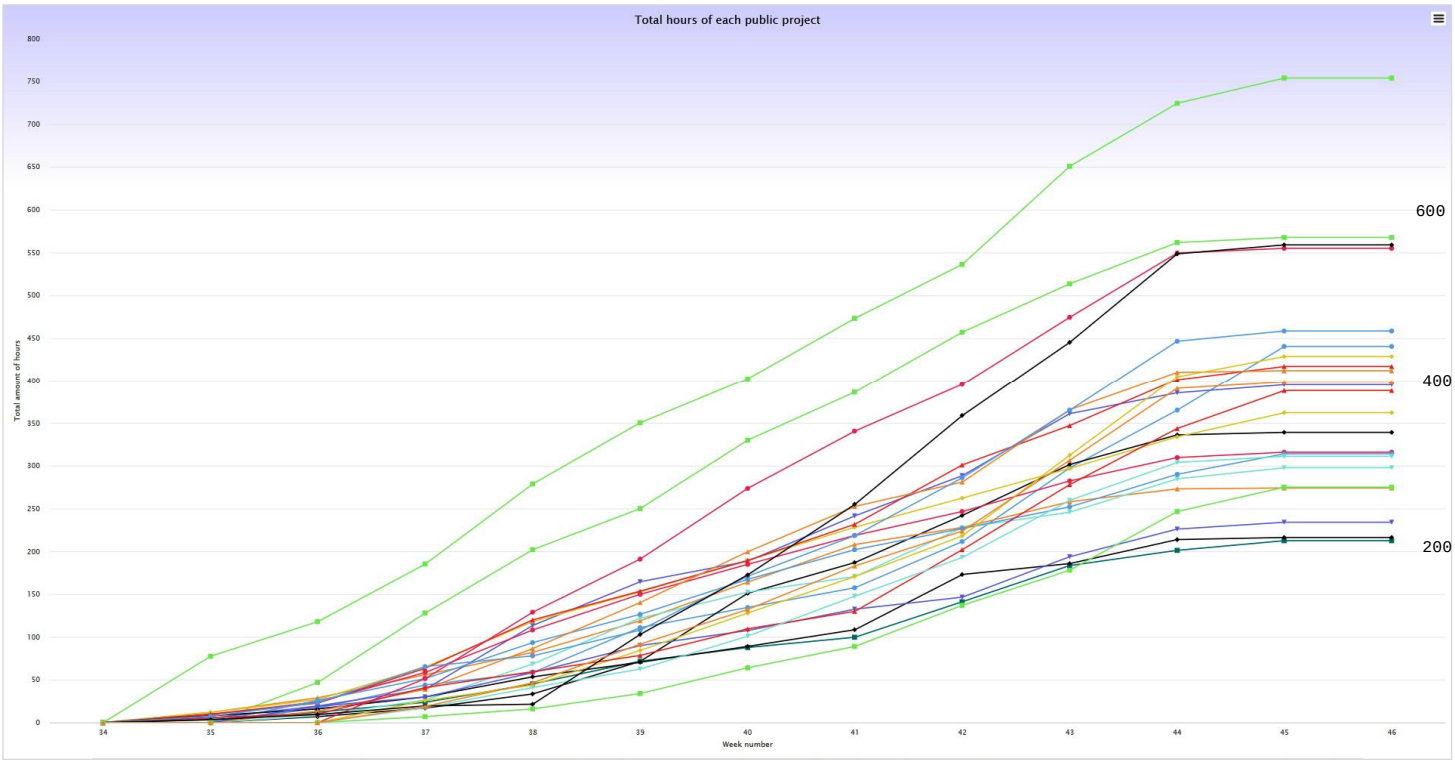
[confluence.atlassian.com]



[i.stack.imgur.com]

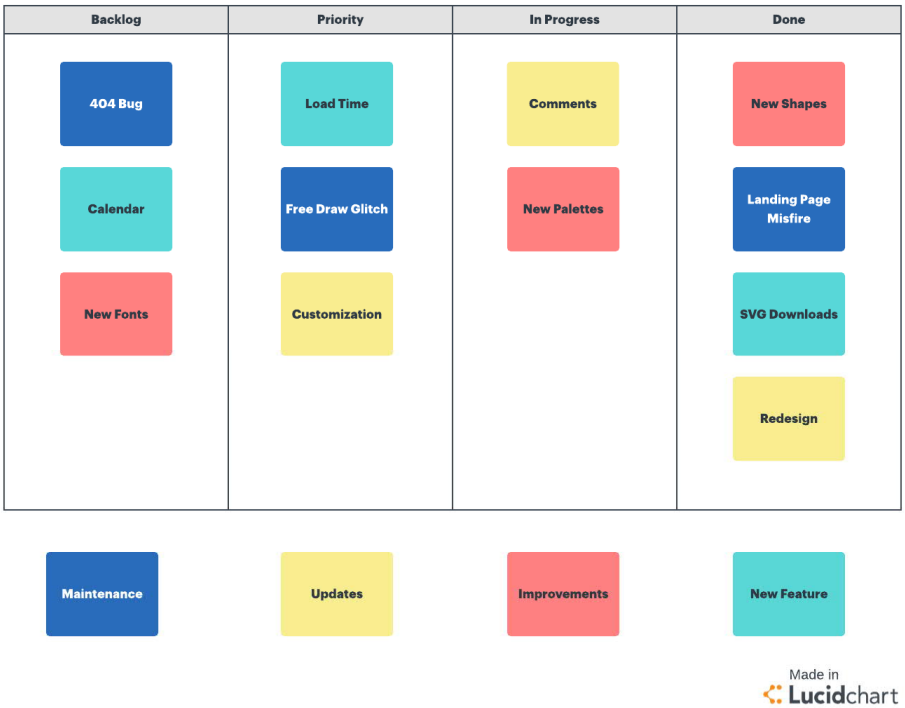
Tampereen yliopisto
Tampere University

One example, PROJ groups, 3/5 Sprints done



Kanban board to make tasks and progress visible

You can use kanban boards in many ways, and for many purposes

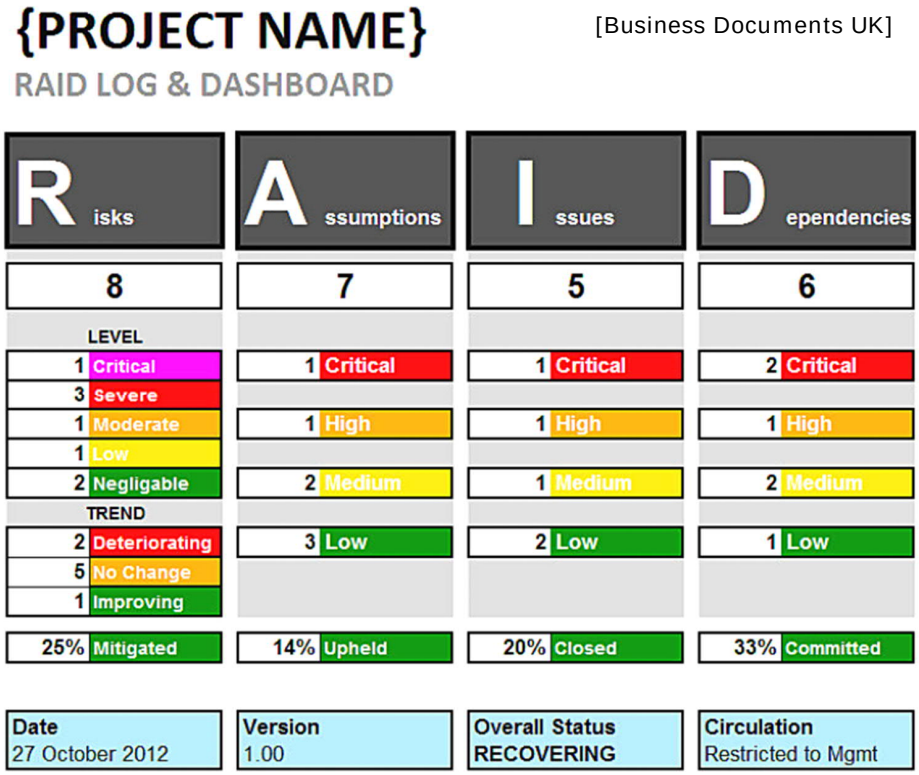


RAID(AR)

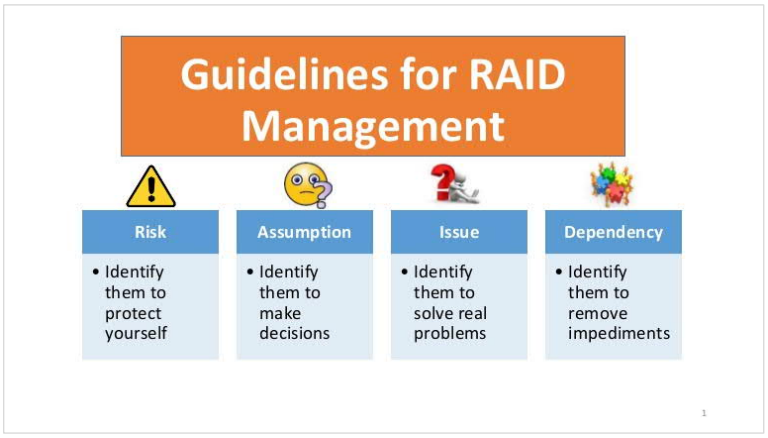
RAID log - RAID is an acronym for risks, assumptions, issues, and dependencies.

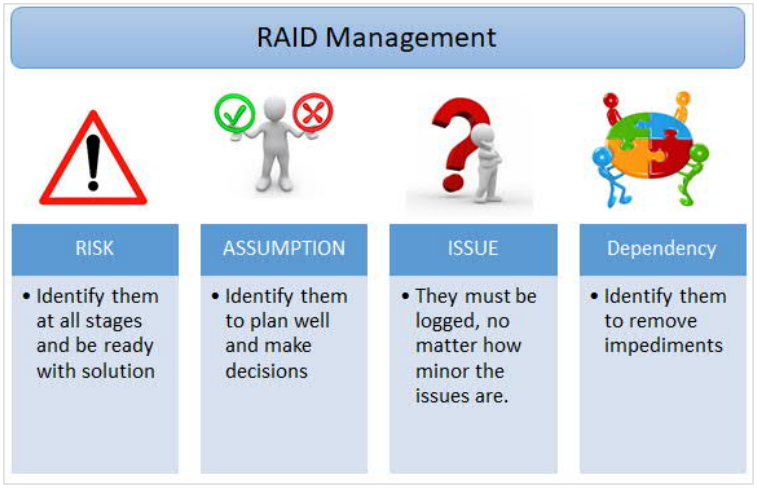
The RAID log is a project management tool that records developments in these four aspects of project work for the stakeholders' benefit and for an end-of-project review.

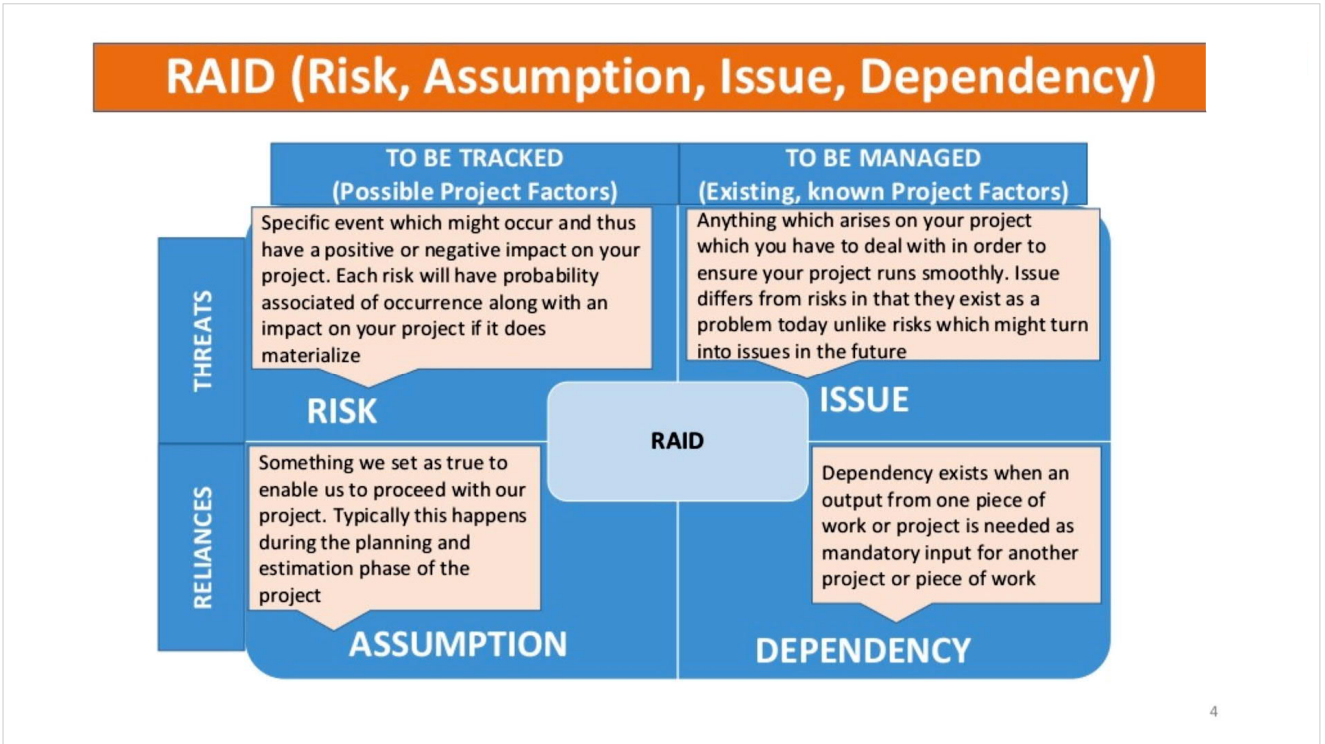
[https://www.smartsheet.com/complete-glossary-project-management-terminology]



RAID as checklist







[https://www.slideshare.net/shrud01/guide-for-raid-management-53629083]

Tampereen yliopisto
Tampere University

[https://www.brightwork.com/blog/raid-management]

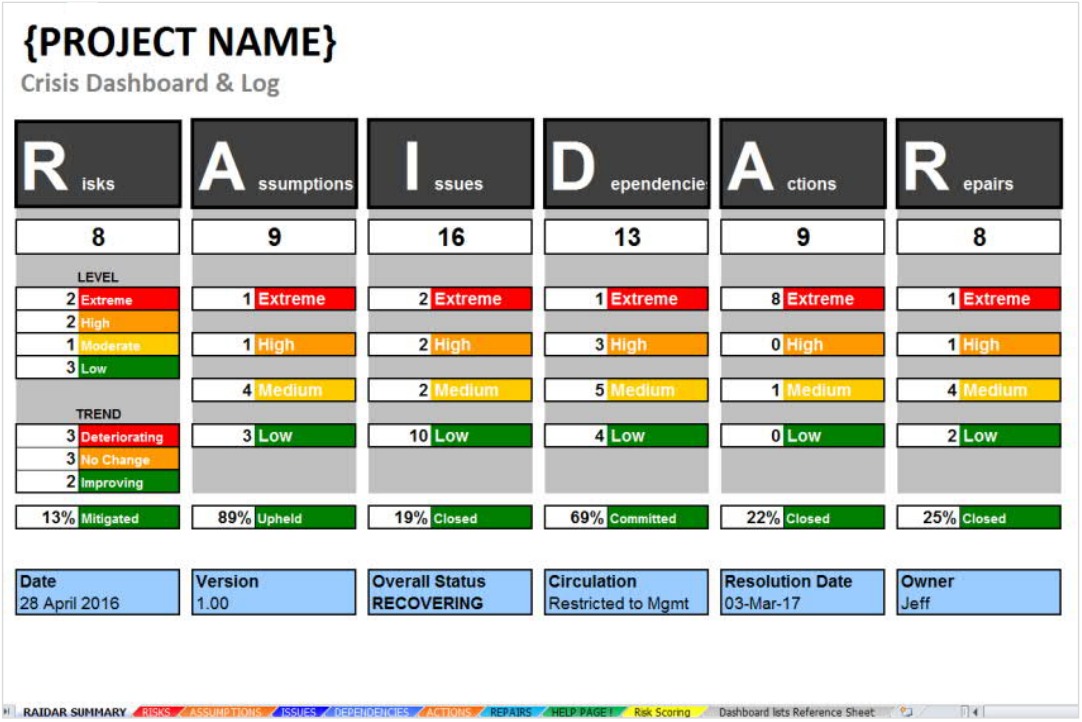
Just another RAID

ONE PAGE PLAN FOR PROJECT RAID SUCCESS

KEY AREAS	KEY NOTES					TARGET STATE	MEASURES
RISKS 	A Risk is defined as uncertainty of outcome, whether positive opportunity or negative threat	Each risk has an associated chance of occurrence along with an impact on your project if it does materialise	If the likelihood of the event happening and impact to the project are both high, the event is classed as a risk	Your project risks are the "issues" waiting to happen	Risk refers to the joint likelihood the event will occur and the impact on the project if it does occur	All risks are identified, documented, analysed and mitigated	All risks identified and mitigated
ASSUMPTIONS 	The definition of an assumption is something that is set as true to enable an organisation to go ahead with a project	A common assumption in many projects is that there will be access to the required specialist resources at the required time	The purpose of tracking assumptions is that you need to be ready for your assumptions being wrong	Any assumptions made about the delivery of externally controlled deliverables should also be included	It is important to include all the assumptions that have identified into a project documents such as a Project Brief	As part of the initial planning phase and project brief, assumptions are identified and documented	No surprises from unidentified or false assumptions
ISSUES 	An issue is something that is going wrong now in a project are those things which has occurred, is current and have yet to be properly addressed	Issues differ from risks in that they exist as a problem today, unlike risks which might turn into issues in the future	Like risks, issues need to be managed through an agreed management process	An issue is something that has gone wrong (deviation from the approved scope, schedule, budget etc.)	Issues are also risks that have been realised, and have turned into issues	All issues are resolved to a positive outcome	All issues are recorded, tracked and have owners
DEPENDENCIES 	Dependencies are something that must be delivered to enable a project's delivery and these must be identified and tracked	Dependencies form a key part of workload priorities during the program and are a basic agenda item for any meetings and decision points	Dependencies exist when an output from one task or another project is needed as a mandatory input for another task or another project	It is a key responsibility for project managers to record, monitor, and manage these dependencies	Dependencies maybe items that are being delivered from elsewhere, and that may not be directly in the control of a project manager	All dependencies are identified, documented and prioritised	No project failure owing to unknown dependencies
RAID LOG 	The RAID log includes a description of each risk, full analysis and a plan to manage it	All risks should have an ID number and an owner	In addition all risks should be measured in a standard way with probability and impact	In the risk log section, there should be columns also for the risk owner, status, and any dates that may be relevant	As with risks and actions, all issues should have an assigned id, owner and timescale	All Risks, Assumptions, Issues and Dependencies logged and managed accordingly	RAID management ensures minimum impact to a successful project outcome

Tampereen yliopisto
Tampere University

[https://business-docs.co.uk/downloads/project-crisis-management-dashboard-and-excel-log-raidar-template/]



{PROJECT NAME}

Crisis Dashboard & Log

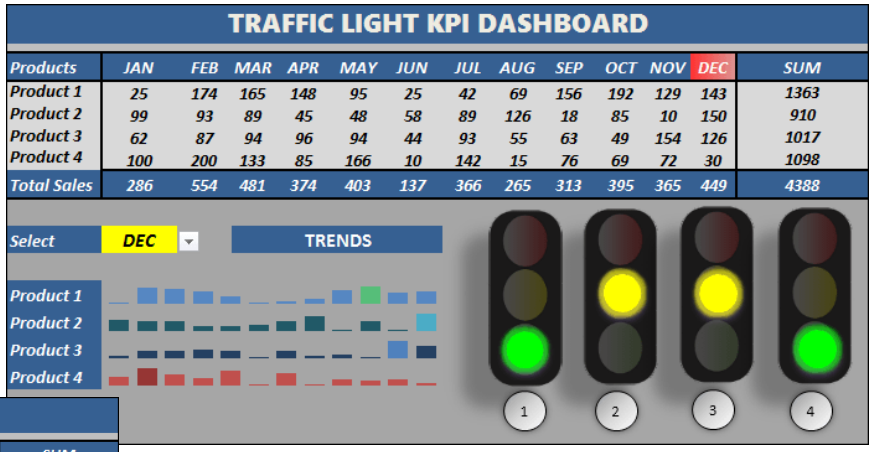
R isks	A ssumptions	I ssues	D ependencie	A ctions	R epairs
8	9	16	13	9	8
LEVEL					
2 Extreme	1 Extreme	2 Extreme	1 Extreme	8 Extreme	1 Extreme
2 High	1 High	2 High	3 High	0 High	1 High
1 Moderate	4 Medium	2 Medium	5 Medium	1 Medium	4 Medium
3 Low	3 Low	10 Low	4 Low	0 Low	2 Low
TREND					
3 Deteriorating					
3 No Change					
2 Improving					
13% Mitigated	89% Upheld	19% Closed	69% Committed	22% Closed	25% Closed

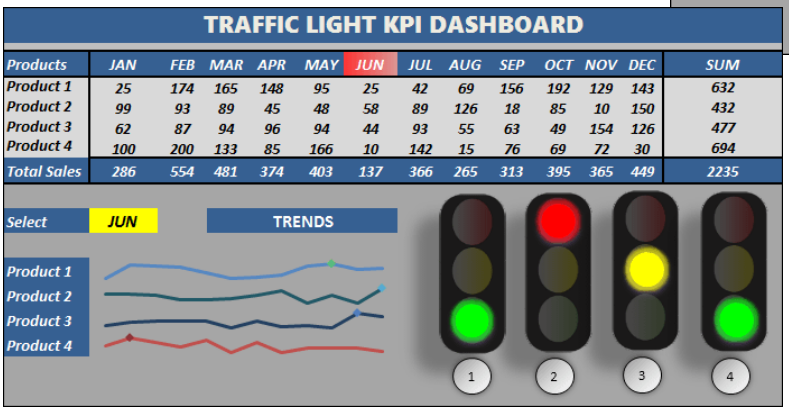
Date	Version	Overall Status	Circulation	Resolution Date	Owner
28 April 2016	1.00	RECOVERING	Restricted to Mgmt	03-Mar-17	Jeff

RAIDAR SUMMARY | RISKS | ASSUMPTIONS | ISSUES | DEPENDENCIES | ACTIONS | REPAIRS | HELP PAGE ! | Risk Scoring | Dashboard lists Reference Sheet

Tampereen yliopisto
Tampere University

Indicators





”Traffic lights” may be used as general indicators of project status. Here based on working hours.

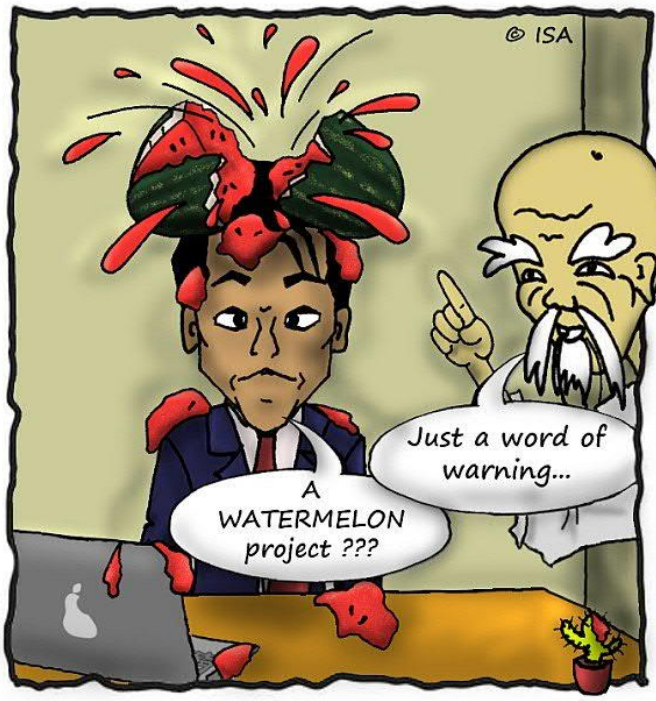
[https://exceldashboardschool.com/excel-traffic-light-dashboard/]

Tampereen yliopisto
Tampere University

Watermelon project

A month before the finish date, everything looks great and all traffic lights for your project are green. And then BOUM! Everything seems to go wrong and suddenly your project is deep in the red. Behind schedule, over cost, a real mess. That’s what’s called a watermelon project.

How to avoid a ”watermelon project”;
- Insist on a clear definition of DONE
- Never accept a percentage of DONE
- Split the tasks to avoid partly done tasks in your project planning.



[https://cactus-competence.com/watermelon-project/]

Tampereen yliopisto
Tampere University

change management

Tampereen yliopisto
Tampere University

change management

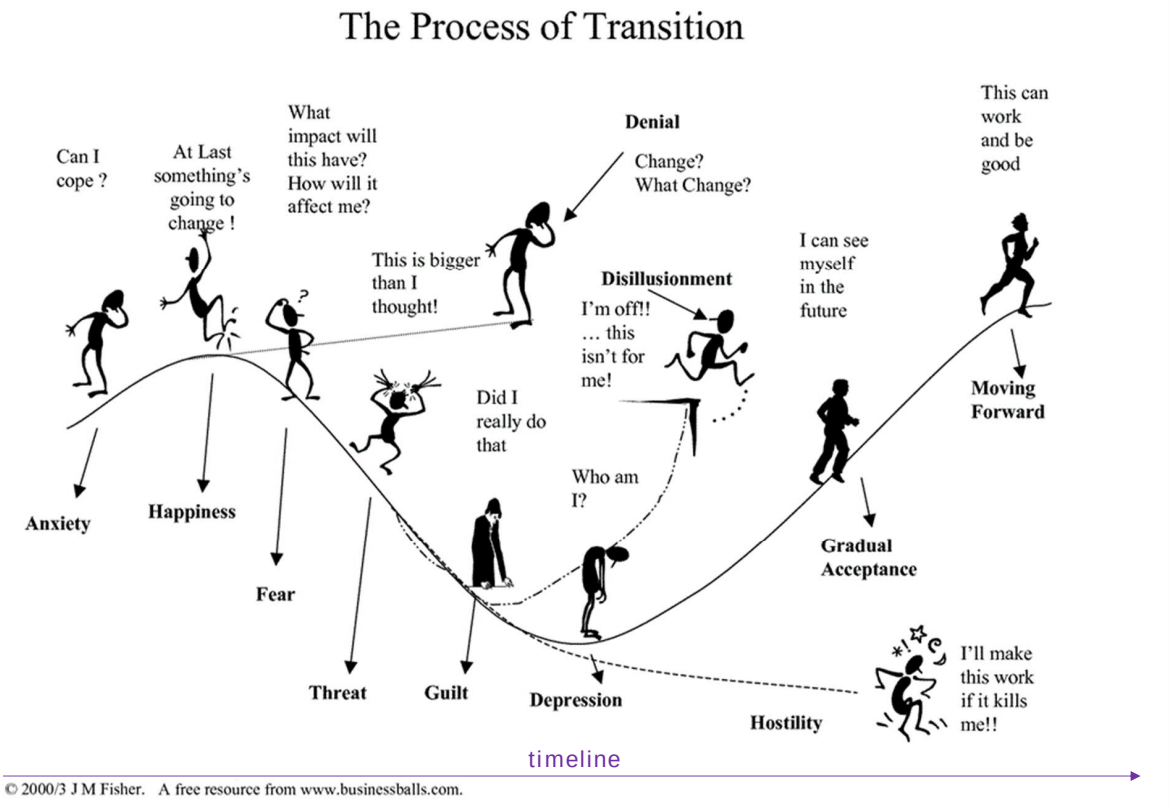
There will be smaller or bigger changes and surprises in every project, so why plan anything at all ?

Surely there will be changes in every project, but the overall goals and techniques are known (planned). Experienced project managers can handle changes.

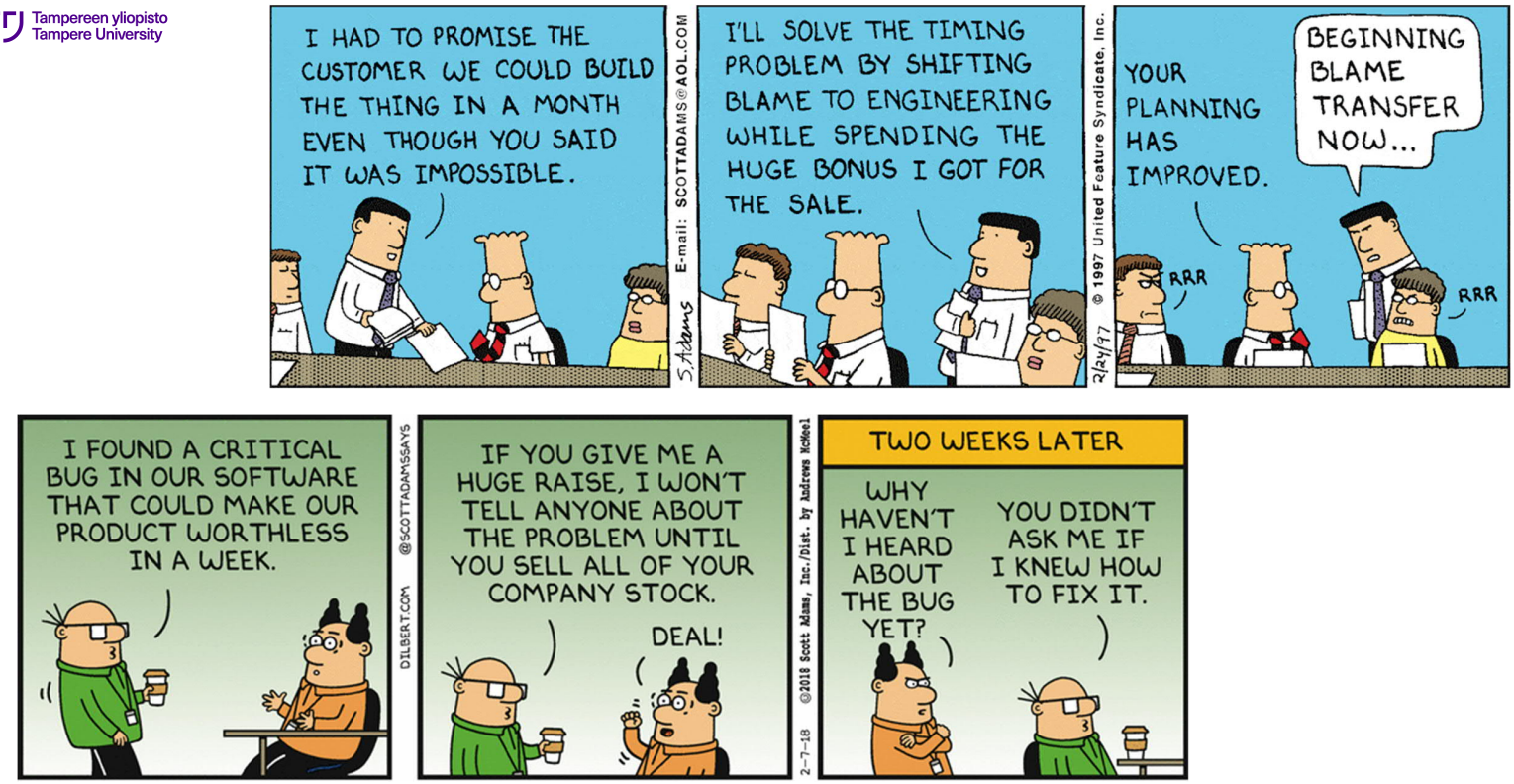
Well, if you change (stretch) the ”iron triangle” (fast – cheap - good), remember to update Project Plan also.

Changes should be documented, and should become known for those who it may concern.

Change


The Process of Transition
Can I cope ?
At Last something's going to change !
What impact will this have? How will it affect me?
This is bigger than I thought!
Denial
Change? What Change?
Disillusionment
I'm off!! … this isn't for me!
Did I really do that
Who am I?
I can see myself in the future
This can work and be good
Moving Forward
Gradual Acceptance
I'll make this work if it kills me!!
Hostility
Depression
Guilt
Threat
Fear
Happiness
Anxiety
timeline
© 2000/3 J M Fisher. A free resource from www.businessballs.com.


WE JUST HAD A MEETING AND DECIDED TO CHANGE YOUR PROJECT SUBSTANTIALLY.
S. Adams E-Mail: SCOTTADAMS@AOL.COM
WE DIDN'T INVITE YOU TO THE MEETING BECAUSE THINGS GO SMOOTHER WHEN NOBODY HAS ANY ACTUAL KNOWLEDGE.
5-21
© 1994 United Feature Syndicate, Inc.
SO, WHAT ARE THE CHANGES?
IF I REMEMBER I'LL LEAVE YOU A VOICE MAIL.



Tampereen yliopisto
Tampere University

risk management

Tampereen yliopisto
Tampere University

risk management activities

[Guide to PMBOK, 2017]

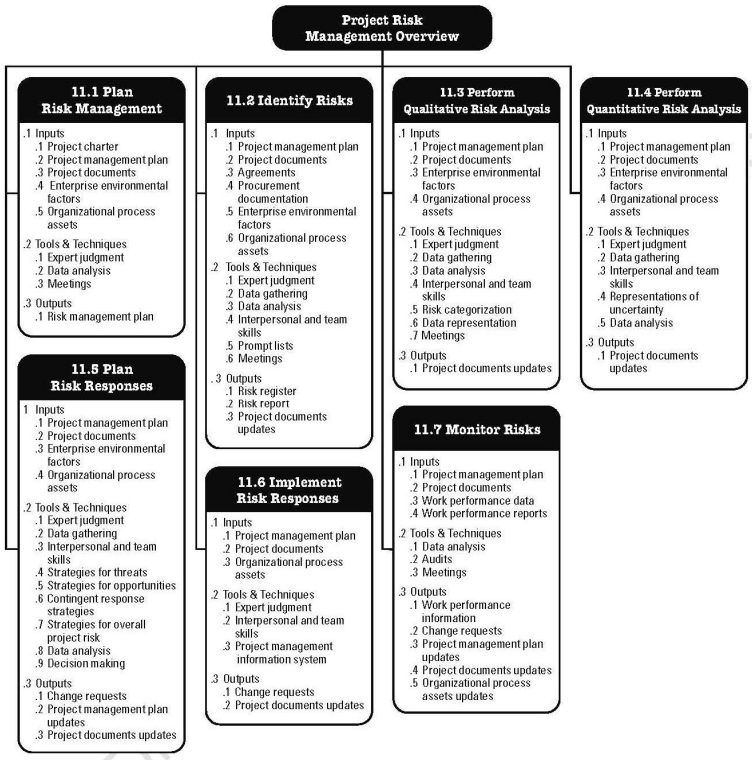


Figure 11-1. Project Risk Management Overview

Tampereen yliopisto
Tampere University

[PMBOK Guide, 2017]

RBS LEVEL 0	RBS LEVEL 1	RBS LEVEL 2
0. ALL SOURCES OF PROJECT RISK	1. TECHNICAL RISK	1.1 Scope definition
		1.2 Requirements definition
		1.3 Estimates, assumptions, and constraints
		1.4 Technical processes
		1.5 Technology
		1.6 Technical interfaces
		Etc.
	2. MANAGEMENT RISK	2.1 Project management
		2.2 Program/portfolio management
		2.3 Operations management
		2.4 Organization
		2.5 Resourcing
		2.6 Communication
		Etc.
	3. COMMERCIAL RISK	3.1 Contractual terms and conditions
		3.2 Internal procurement
		3.3 Suppliers and vendors
		3.4 Subcontracts
		3.5 Client/customer stability
		3.6 Partnerships and joint ventures
		Etc.
	4. EXTERNAL RISK	4.1 Legislation
		4.2 Exchange rates
		4.3 Site/facilities
		4.4 Environmental/weather
		4.5 Competition
		4.6 Regulatory
		Etc.

Figure 11-4. Extract from Sample Risk Breakdown Structure (RBS)

risk management

Table 11-1. Example of Definitions for Probability and Impacts

SCALE	PROBABILITY	+/− IMPACT ON PROJECT OBJECTIVES		
		TIME	COST	QUALITY
Very High	>70%	>6 months	>$5M	Very significant impact on overall functionality
High	51-70%	3-6 months	$1M-$5M	Significant impact on overall functionality
Medium	31-50%	1-3 months	$501K-$1M	Some impact in key functional areas
Low	11-30%	1-4 weeks	$100K-$500K	Minor impact on overall functionality
Very Low	1-10%	1 week	<$100K	Minor impact on secondary functions
Nil	<1%	No change	No change	No change in functionality

[Guide to PMBOK, 2017]

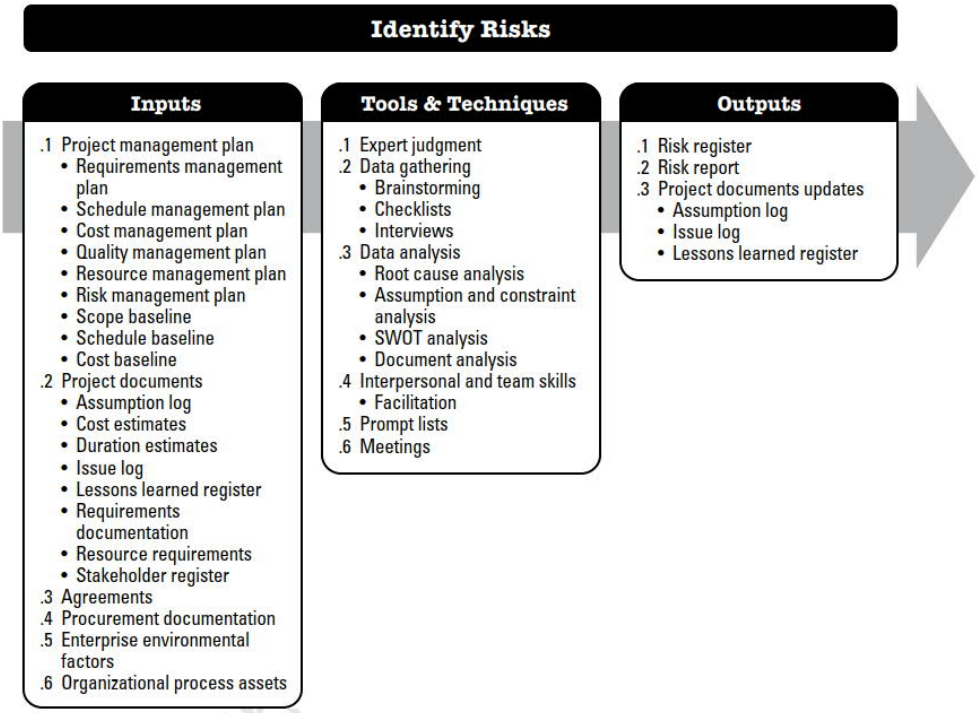
risk management

Probability	Threats					Opportunities					Probability
Very High 0.90	0.05	0.09	0.18	0.36	0.72	0.72	0.36	0.18	0.09	0.05	Very High 0.90
High 0.70	0.04	0.07	0.14	0.28	0.56	0.56	0.28	0.14	0.07	0.04	High 0.70
Medium 0.50	0.03	0.05	0.10	0.20	0.40	0.40	0.20	0.10	0.05	0.03	Medium 0.50
Low 0.30	0.02	0.03	0.06	0.12	0.24	0.24	0.12	0.06	0.03	0.02	Low 0.30
Very Low 0.10	0.01	0.01	0.02	0.04	0.08	0.08	0.04	0.02	0.01	0.01	Very Low 0.10
	Very Low 0.05	Low 0.10	Moderate 0.20	High 0.40	Very High 0.80	Very High 0.80	High 0.40	Moderate 0.20	Low 0.10	Very Low 0.05	
	Negative Impact					Positive Impact					

Figure 11-5. Example Probability and Impact Matrix with Scoring Scheme

[Guide to PMBOK, 2017]

First identify risks



[PMBOK Guide, 2017]

Figure 11-6. Identify Risks: Inputs, Tools & Techniques, and Outputs

7 common project management risks in software development [https://tsh.io/blog/it-project-risk-management/]

- Changing requirements and priorities
- Lack of commitment
- Lack of communication
- Poor documentation
- Unplanned absence of a team member
- Poor communication with the client
- Failure to deliver on time.

There is no software project without risks.

- hold regular meetings
- address problems immediately
- search for, share, document and understand knowledge and data
- motivate all team members within your organization.

Tampereen yliopisto
Tampere University

10 biggest risks in software development?
Mikaela Robertson, 08 July 2020

1. Inaccurate Estimations
2. Scope Variations
3. End-user Engagement
4. Stakeholder Expectations
5. Poor Quality Code
6. Poor Productivity
7. Inadequate Risk Management
8. Low Stakeholder Engagement
9. Inadequate Human Resources
10. Lack of Ownership.

[https://codebots.com/library/way-of-working/what-are-the-10-biggest-risks-in-software-development]

Tampereen yliopisto
Tampere University

One risk, it's management, 1/2

Common risk sources

- project group (skills, sickness, motivation, burnout,…)
- customer (commitment, busy times, understanding,…)
- technology (hw, sw, APIs, tools,…)
- environment, 3rd party, "outsiders" (competitors, licences, subcontractors,..).

Usually risks are evaluated by scale 1..3, no need for more numbers.

Risk importance, significance, seriousness, exposure = severity (impact, effect) * probability (likelihood) i.e.
risk importance = severity * probability

One risk, it's management, 2 / 2

About each (significant, no time for all) risk you have better consider

- risk ID (unique "name")
- probability (e.g. 1..2..3, biggest number means high ?)
- severity (e.g. 1..2..3, biggest number means high ?)
- early warning signs (e.g. low motivation; missing meetings, no replies)
- how to avoid (so that risk do not realise)
- how to recover (if risk has already realised).

Always have a "Plan B", or several such workarounds.

Prepare for surprises, have buffer time at schedule (20 % ?), double resources, back-up plans.

That may be used as a checklist of matters which affect risks

298 Software Development Project Risk Management: A New Conceptual Framework

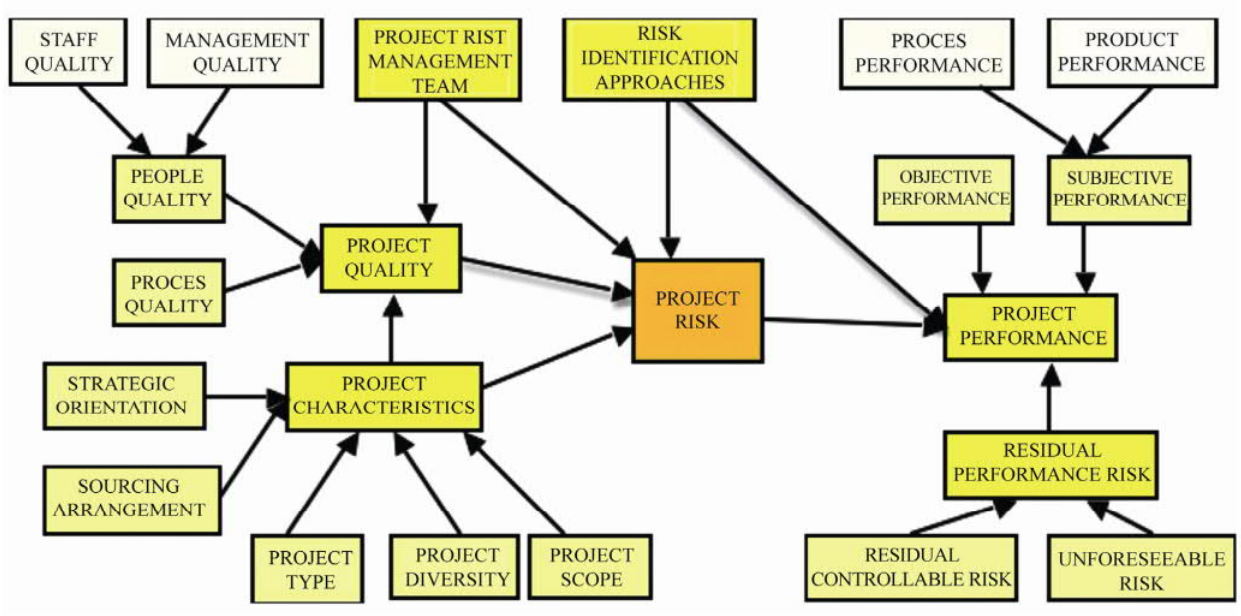


Figure 1. Conceptual framework.

Tampereen yliopisto
Tampere University

By the way, in business risk evaluation, many companies say that 50 % risk is not worth trying.

So, what you mean by ”50 % risk” ?

Risk Matrix

Impact	Probability				
	Rare (0 - 20%)	Unlikely (20 - 40%)	Possible (40 - 60%)	Likely (60 - 80%)	Almost Certain(>80%)
Catastrophic			1(0*)		
Major				1(1*)	1(0*)
Moderate					1(0*)
Minor		1(0*)			
Insignificant					

Total Risks : 9 / Open : 4 / Reviewed : 4 / Closed : 1 (*Closed Risks)

[http://zilicus.com/features/project-risk-management.html]

Tampereen yliopisto
Tampere University

Just another risk evaluation matrix

Risk Matrix Heat Map

	Rare	Unlikely	Possible	Likely	Almost Certain
Insignificant	Low	Low	Low	Low	Low
Minor	Low	Low	Low	Medium	Medium
Moderate	Low	Low	Medium	Medium	Medium
Major	Low	Medium	Medium	High	High
Extreme	Low	Medium	Medium	High	Extreme

Category 1
Category 2
Category 3
Category 4

[https://www.slideteam.net/business_powerpoint_diagrams/]

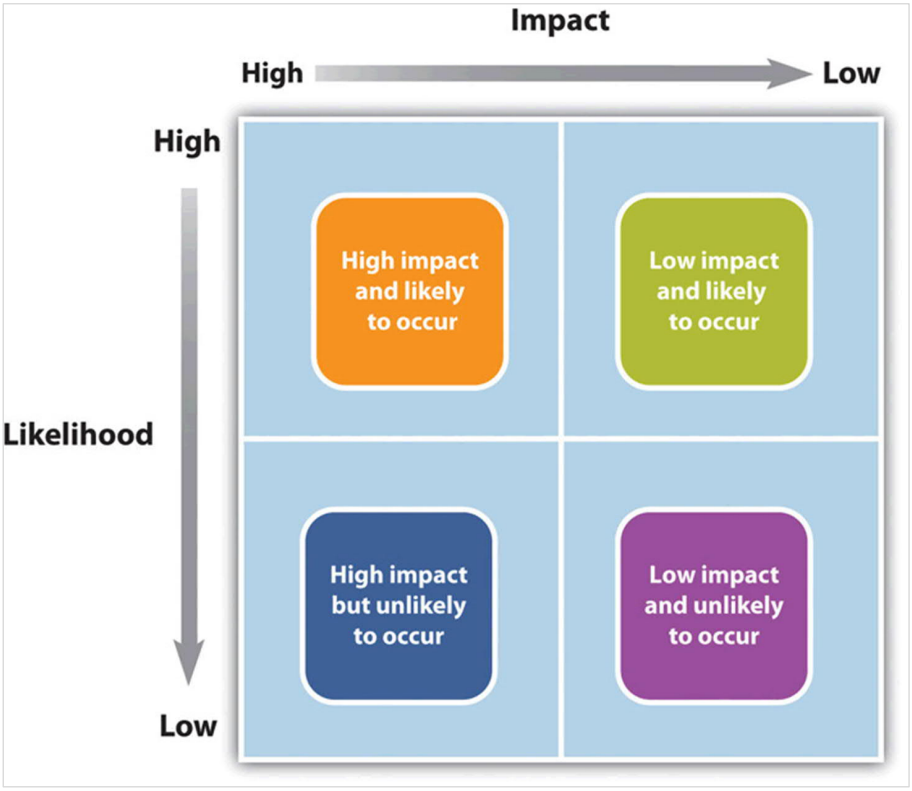
Tampereen yliopisto
Tampere University

One indicator of project risk management success is, if at final report there are not any unforeseen risks.

”Our risk management was successful, as we encountered all risks we listed.”

Tampereen yliopisto
Tampere University

Risk and impact (just another scale)



[https://pm4id.org/chapter/11-2-risk-management-process/]

PROJ example risk management list

METRICS
MONITORING
TOOL

PROJ-H2020-G

Scale

VL = 1
L = 2
M = 3
H = 4
VH = 5

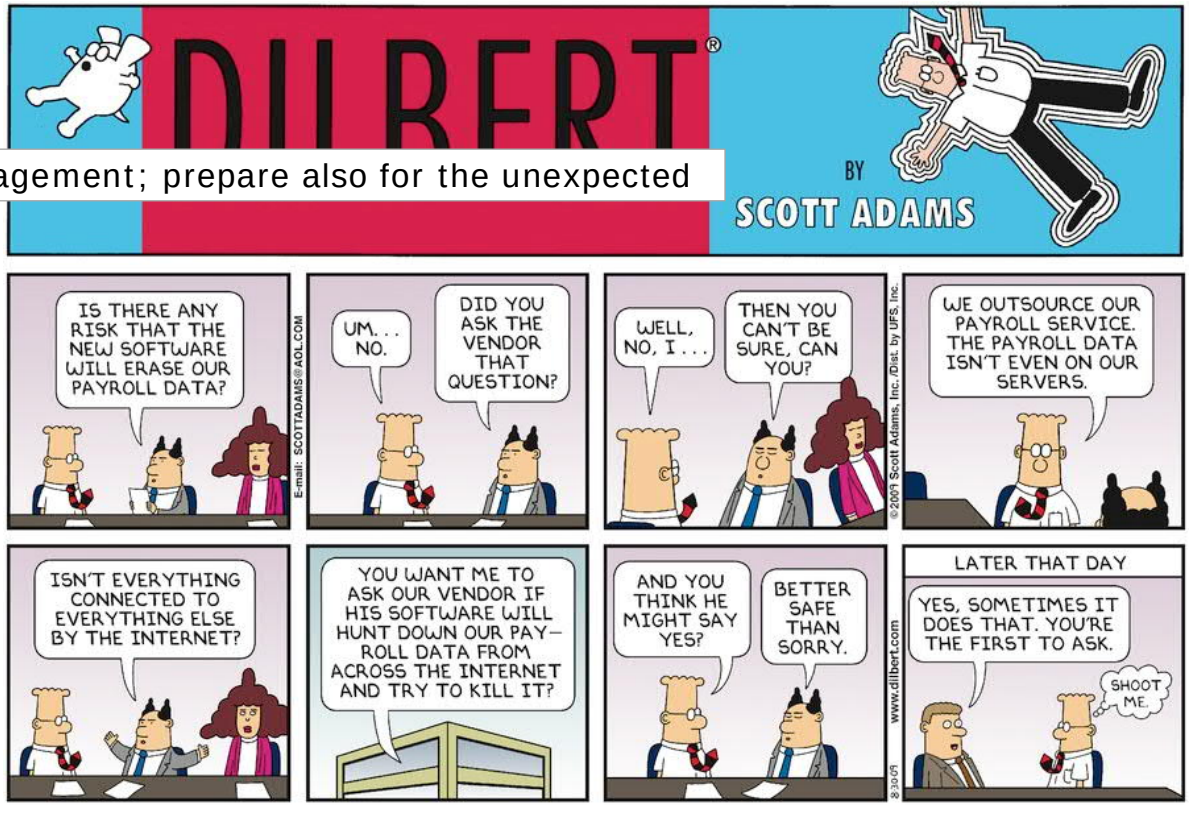
Project Info | Members | Reports | Log Time | Risks | Charts

Project Risks

Risk	Impact	Probability	Actions	
Customer stops participating in the project	Very High	Very Low	Edit	5*1=5
Lack of time among project members	High	Medium	Edit	4*3=12
Amount of requirements gets too high	Medium	Low	Edit	3*2=6
Key group member leaves	High	Low	Edit	4*2=8
Lack of experience with django	Medium	Medium	Edit	3*3=9
Slow start	Low	Very High	Edit	2*5=10
Architect overworked	Medium	Very High	Edit	3*5=15
Customer overworked	High	Very Low	Edit	4*1=4
Lack of motivation among members	Medium	Medium	Edit	3*3=9
Most work on weekends sprint review Monday	Medium	High	Edit	3*4=12

Calculations made afterwards by Tensu

Risk management; prepare also for the unexpected



Standards


SFS
Projektinhallinnan ISO-standardit
Project management standards
1. ISO 21500:2012 Guidance on project management
2. ISO 21503:2017 Guidance on programme management
3. ISO 21504:2015 Guidance on portfolio management
4. ISO 21505:2017 Guidance on governance
5. ISO 21508:2018 Earned value management in project and programme management
6. ISO 21511:2018 Work breakdown structures for project and programme management
Suomen Standardisoimisliitto
12 04.12.2019 Janne Kalli

Tampereen yliopisto
Tampere University

Notice that standards are like checklists, they do not give detailed help to your project.

SUOMEN STANDARDISOIMISLIITTO SFS | STANDARDI | **SFS-ISO 21500**

Suomen Standardisoimisliitto SFS
Finnish Standards Association SFS

Vahvistettu
2012-10-08

1 (1 + 36)

SFS/ICS 03.100

Tämä standardi on vahvistettu englanninkielisenä | *This standard is approved in English*

GUIDANCE ON PROJECT MANAGEMENT

Tämä standardi sisältää kansainvälisen standardin ISO 21500:2012 "Guidance on project management" englanninkielisen tekstin.

Kansainvälinen standardi ISO 21500:2012 on vahvistettu suomalaiseksi kansalliseksi standardiksi.

This standard consists of the English text of the International Standard ISO 21500:2012 "Guidance on project management".

The International Standard ISO 21500:2012 has the status of a Finnish national standard.

Tampereen yliopisto
Tampere University

[ISO 21500:2012 Guidance on project management]

Contents

Page

Project management helps business.

[ISO 21500:2012]

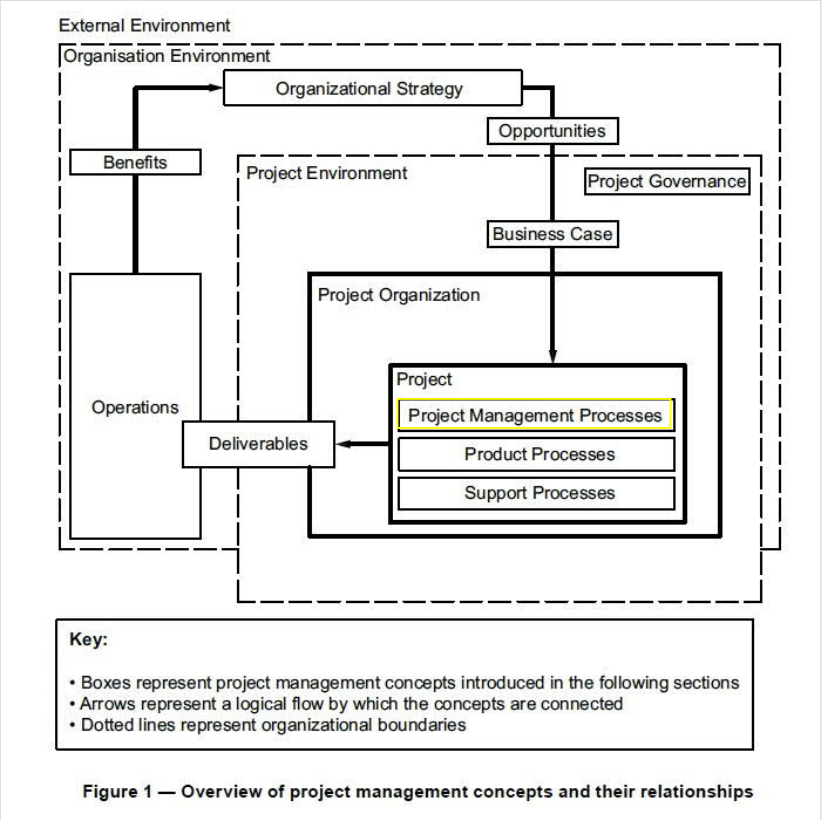
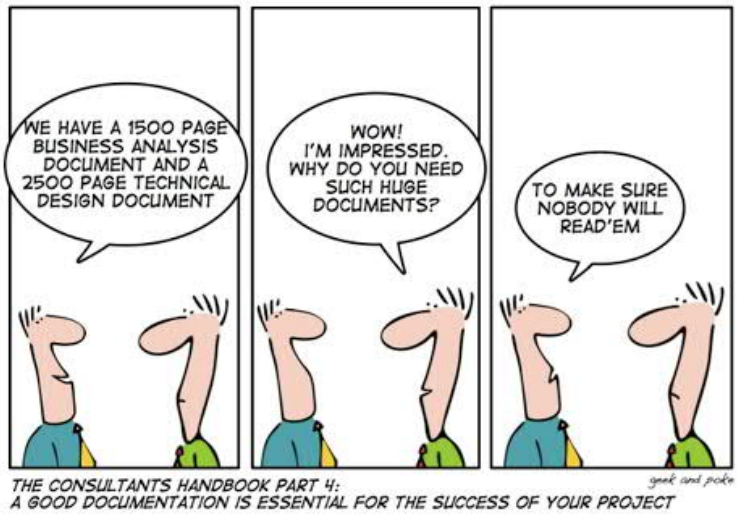


Figure 1 — Overview of project management concepts and their relationships



Day four of the annual "Mac/PC Standardization Convention".

Tampereen yliopisto
Tampere University

ISO/IEC 25010:2011(E)

Product quality

These quality attributes should be considered in software projects.

But how to measure those ?

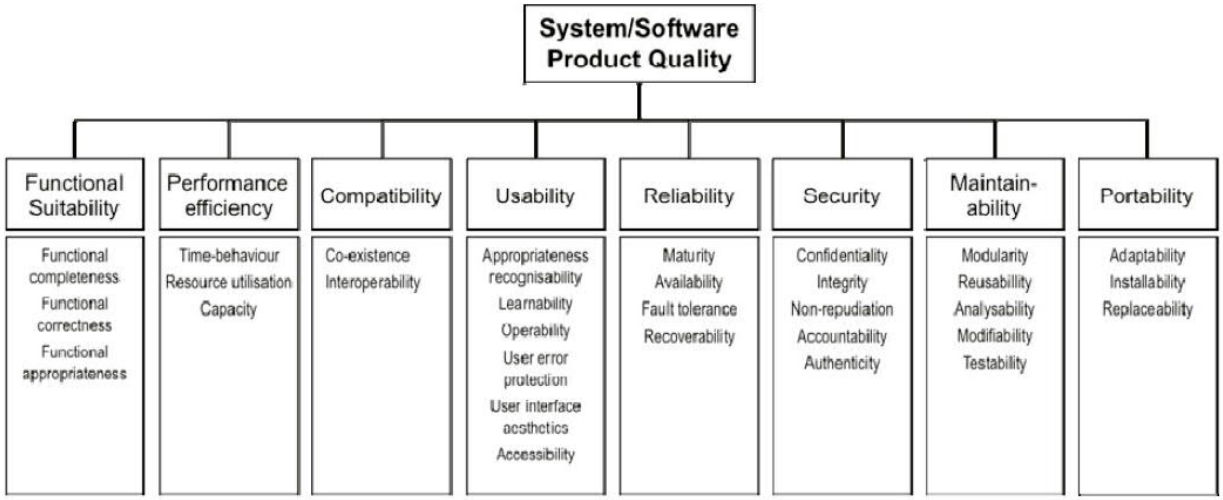


Figure 4 — Product quality model

Tampereen yliopisto
Tampere University

Product quality, in detail 1 / 2

Product Quality - ISO/IEC 25010

Characteristics	Sub-Characteristics	Definition
Functional Suitability	Functional Completeness	degree to which the set of functions covers all the specified tasks and user objectives.
	Functional Correctness	degree to which the functions provides the correct results with the needed degree of precision.
	Functional Appropriateness	degree to which the functions facilitate the accomplishment of specified tasks and objectives.
Performance Efficiency	Time-behavior	degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.
	Resource Utilization	degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.
	Capacity	degree to which the maximum limits of the product or system, parameter meet requirements.
Compatibility	Co-existence	degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.
	Interoperability	degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.
Usability	Appropriateness recognisability	degree to which users can recognize whether a product or system is appropriate for their needs.
	Learnability	degree to which a product or system enables the user to learn how to use it with effectiveness, efficiency in emergency situations.
	Operability	degree to which a product or system is easy to operate, control and appropriate to use.
	User error protection	degree to which a product or system protects users against making errors.
	User interface aesthetics	degree to which a user interface enables pleasing and satisfying interaction for the user.
	Accessibility	degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.

Tampereen yliopisto
Tampere University

Product quality, in detail 2 / 2 [ISO 25010]

Reliability	Maturity	degree to which a system, product or component meets needs for reliability under normal operation.
	Availability	degree to which a product or system is operational and accessible when required for use.
	Fault tolerance	degree to which a system, product or component operates as intended despite the presence of hardware or software faults.
	Recoverability	degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.
Security	Confidentiality	degree to which the prototype ensures that data are accessible only to those authorized to have access.
	Integrity	degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.
	Non-repudiation	degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later.
	Accountability	degree to which the actions of an entity can be traced uniquely to the entity.
	Authenticity	degree to which the identity of a subject or resource can be proved to be the one claimed.
Maintainability	Modularity	degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.
	Reusability	degree to which an asset can be used in more than one system, or in building other assets.
	Analyzability	degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
	Modifiability	degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.
	Testability	degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.
Portability	Adaptability	degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.
	Installability	degree of effectiveness and efficiency in which a product or system can be successfully installed and/or uninstalled in a specified environment.
	Replaceability	degree to which a product can replace another specified software product for the same purpose in the same environment.

Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

Highlights - What to remember

- project work IS teamwork
- problems at projects are seldom technical (”hard”), but ”soft”
- agree project guidelines, regulations and rules, and follow them
- somehow get customer to be committed to the project
- ask help from your teammates, also help them when needed
- remember professional ethics (avoid shortcuts = technical debt)
- be honest, give constructive feedback
- if you see problems in project, tell your superior/boss about them, don’t hide
- understand that sometimes there are ”slow flow” on your or groupmember’s work
- sometimes it is good to use humour in difficult situations (at least in Finland).

”Workload is not killing, it is challenging.”

”Student’s life is not always miserable, sometimes it is just a bad day.”

Tampereen yliopisto
Tampere University

Highlights - What to remember

- think WHAT you need to know or monitor in a project, and measure that wise
- use reasonable metrics (= right things, just enough, not too much)
- estimations are estimations, good measurements are exact
- use historical data; from previous projects and/or your experiences, ask seniors
- best basis for estimations is experience… which can not be obtained by reading books nor from lectures, you have to work and collect your own data
- estimating size or effort, feel free to ask senior colleagues for their opinion
- discuss in group about size or effort estimations
- you may start estimating of modules/components e.g. ”is this bigger or smaller what have been done earlier”
- as a project manager, remember to delegate (you SHOULD delegate some tasks)
- coffee room table is an important project management tool (= discussions). ;-)

Tampereen yliopisto
Tampere University

Now the additional L10 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Now the additional L10 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Now the additional L10 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Now the additional L10 extra slides are here

No time to show these at lectures, but otherwise good to know, at least if you are a major reader.

Tampereen yliopisto
Tampere University

Tampereen yliopisto
Tampere University

Overview of a project management

Tampereen yliopisto
Tampere University

How is YOUR project going on ?

[Agile product management, 2010]

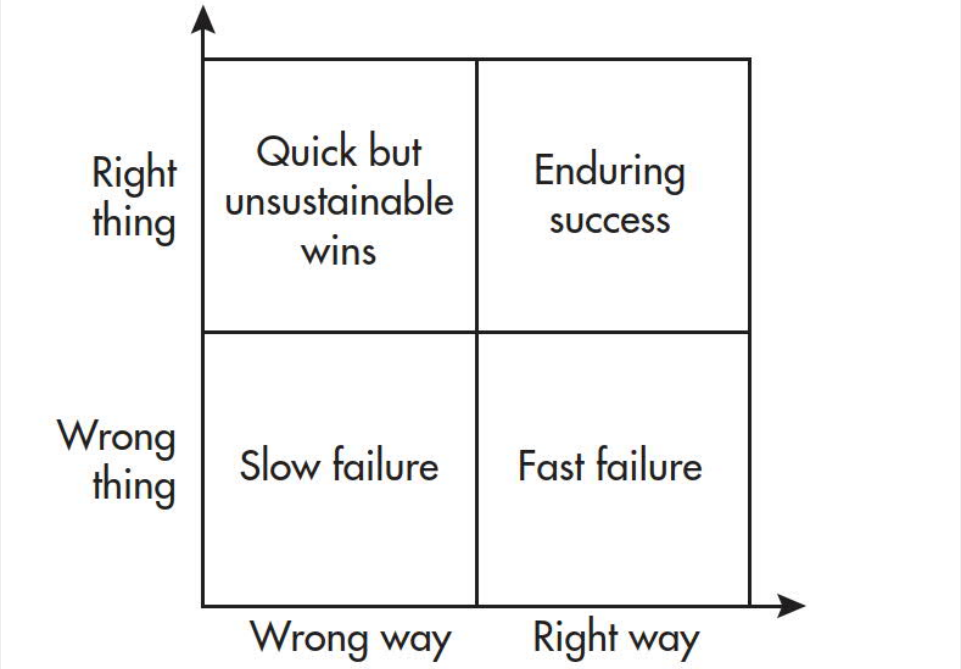


FIGURE 1.1 Doing the right thing the right way

Universal management activities

✧ *Project planning*

- Project managers are responsible for planning. estimating and scheduling project development and assigning people to tasks.
- Covered in Chapter 23.

✧ *Risk management*

- Project managers assess the risks that may affect a project, monitor these risks and take action when problems arise.

✧ *People management*

- Project managers have to choose people for their team and establish ways of working that leads to effective team performance.

Management activities

✧ *Reporting*

- Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software.

✧ *Proposal writing*

- The first stage in a software project may involve writing a proposal to win a contract to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out.

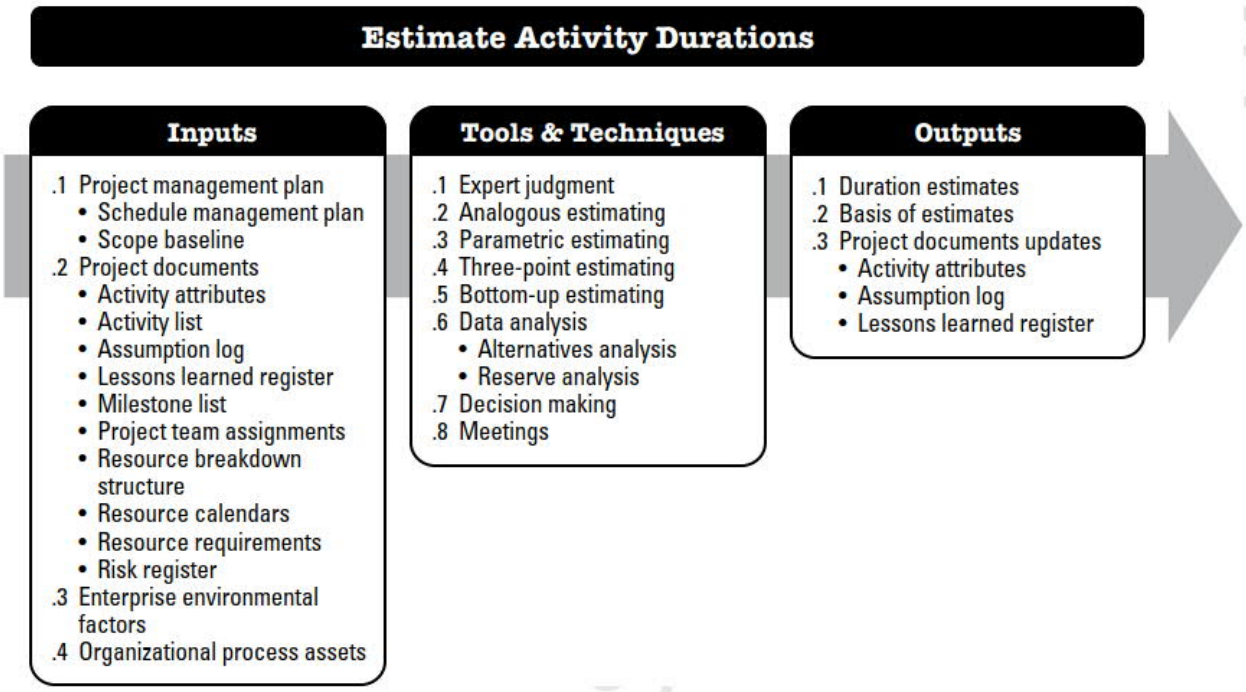


Figure 6-12. Estimate Activity Durations: Inputs, Tools & Techniques, and Outputs

[PMBOK Guide 2017]

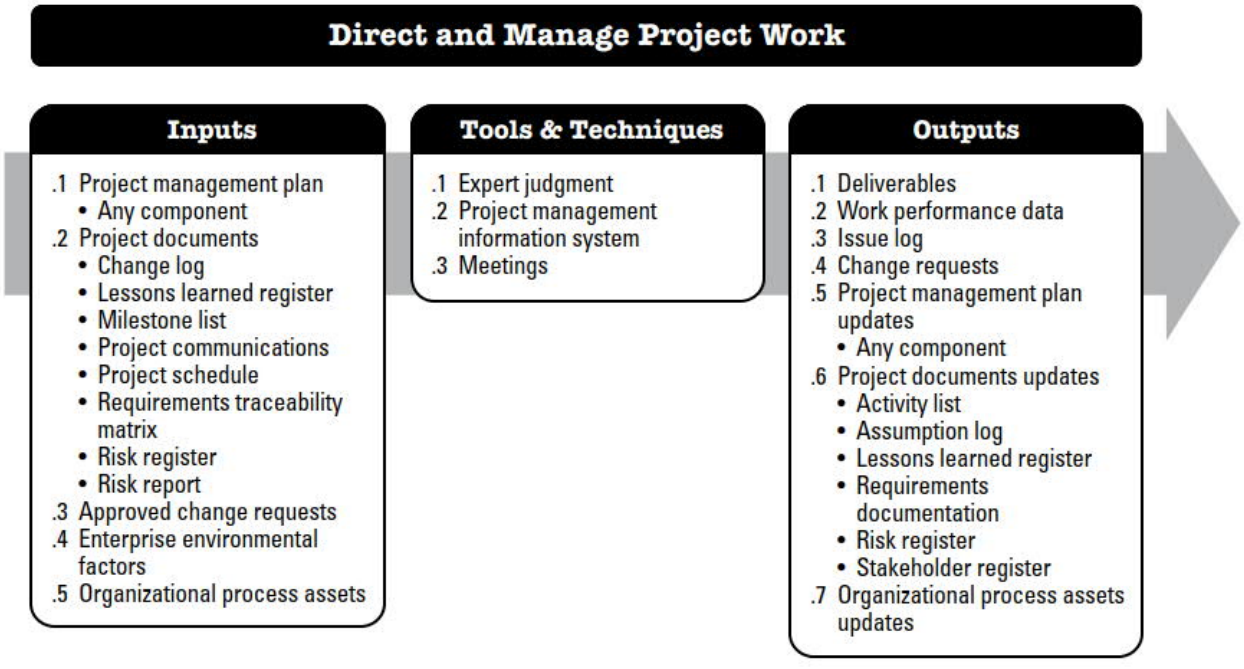


Figure 4-6. Direct and Manage Project Work: Inputs, Tools & Techniques, and Outputs

[PMBOK Guide 2017]

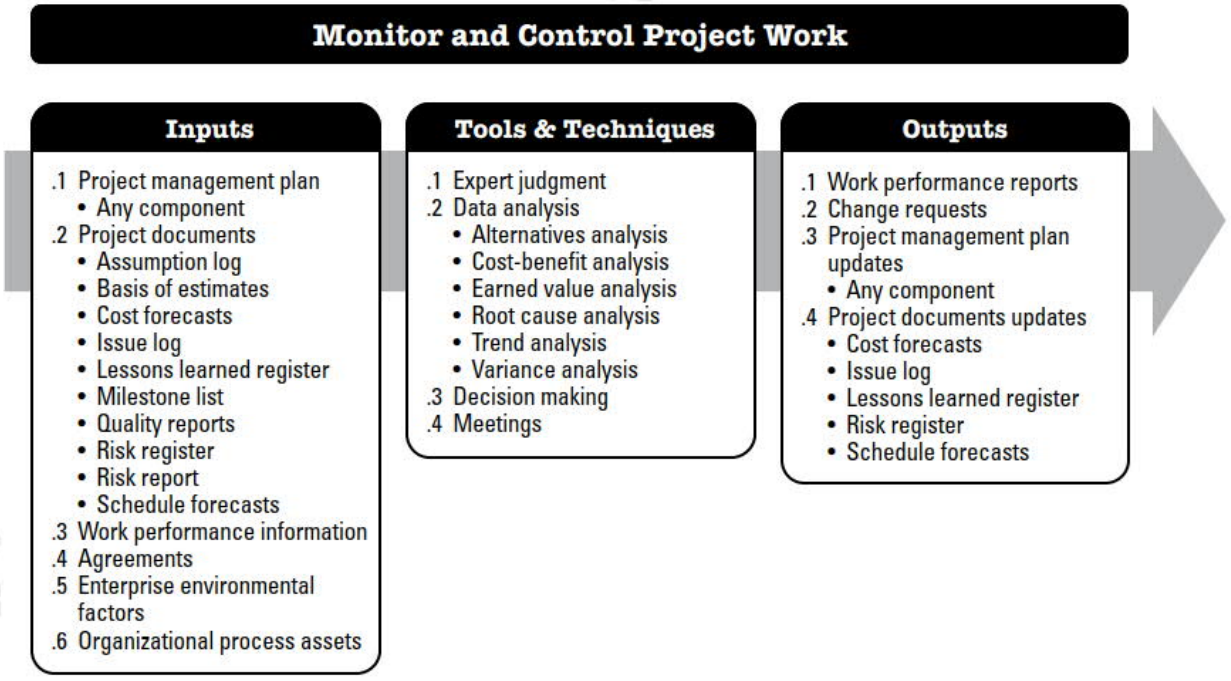


Figure 4-10. Monitor and Control Project Work: Inputs, Tools & Techniques, and Outputs

[PMBOK Guide 2017]

Project scope management

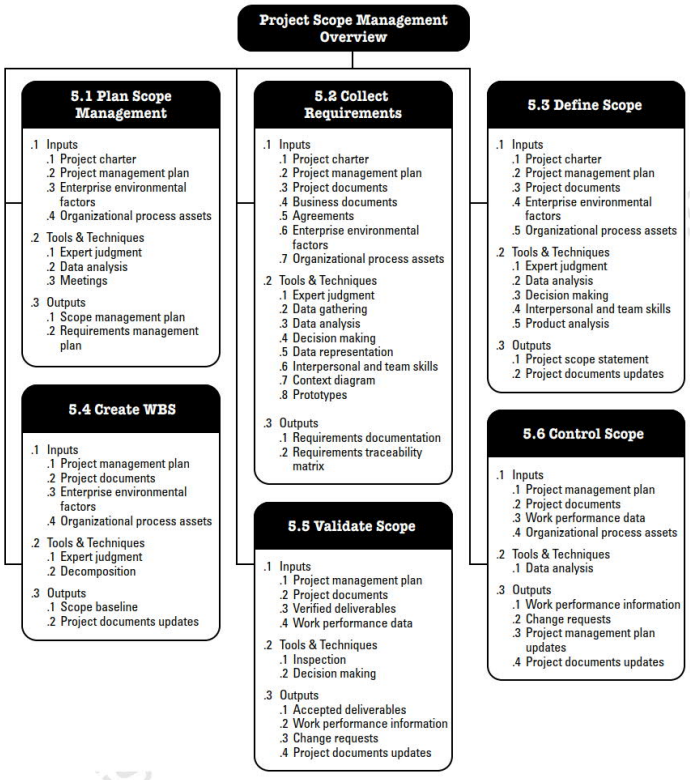
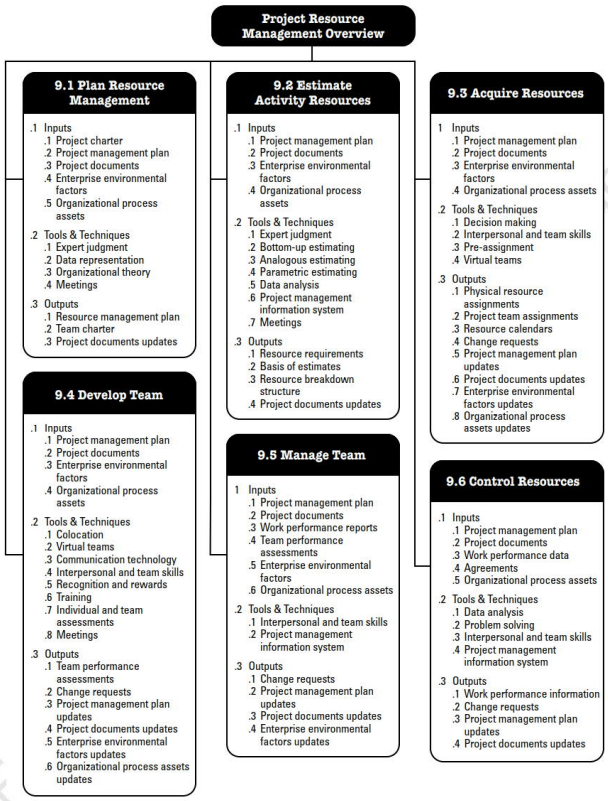


Figure 5-1. Project Scope Management Overview

[PMBOK Guide 2017]

Tampereen yliopisto
Tampere University

Project resource management

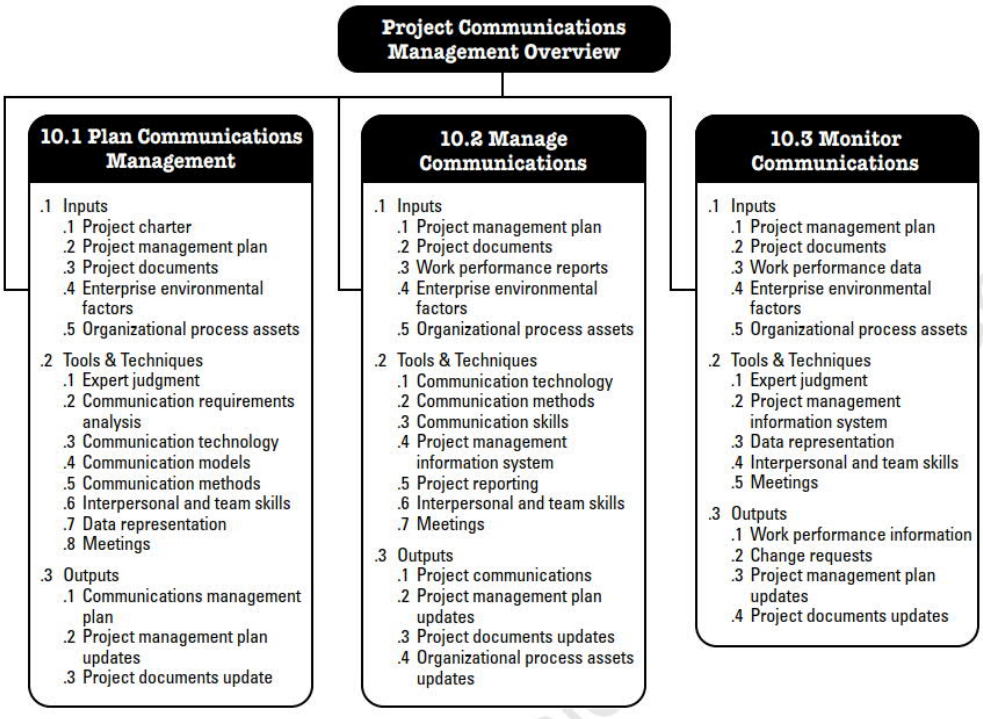


[PMBOK Guide 2017]

Figure 9-1. Project Resource Management Overview

Tampereen yliopisto
Tampere University

Project communications management



[PMBOK Guide 2017]

Figure 10-1. Project Communications Overview

6point6 commissioned a survey of 300 CIOs in the UK and the US to examine their experiences of agile and measure how successfully the principles of agile are being applied and executed.

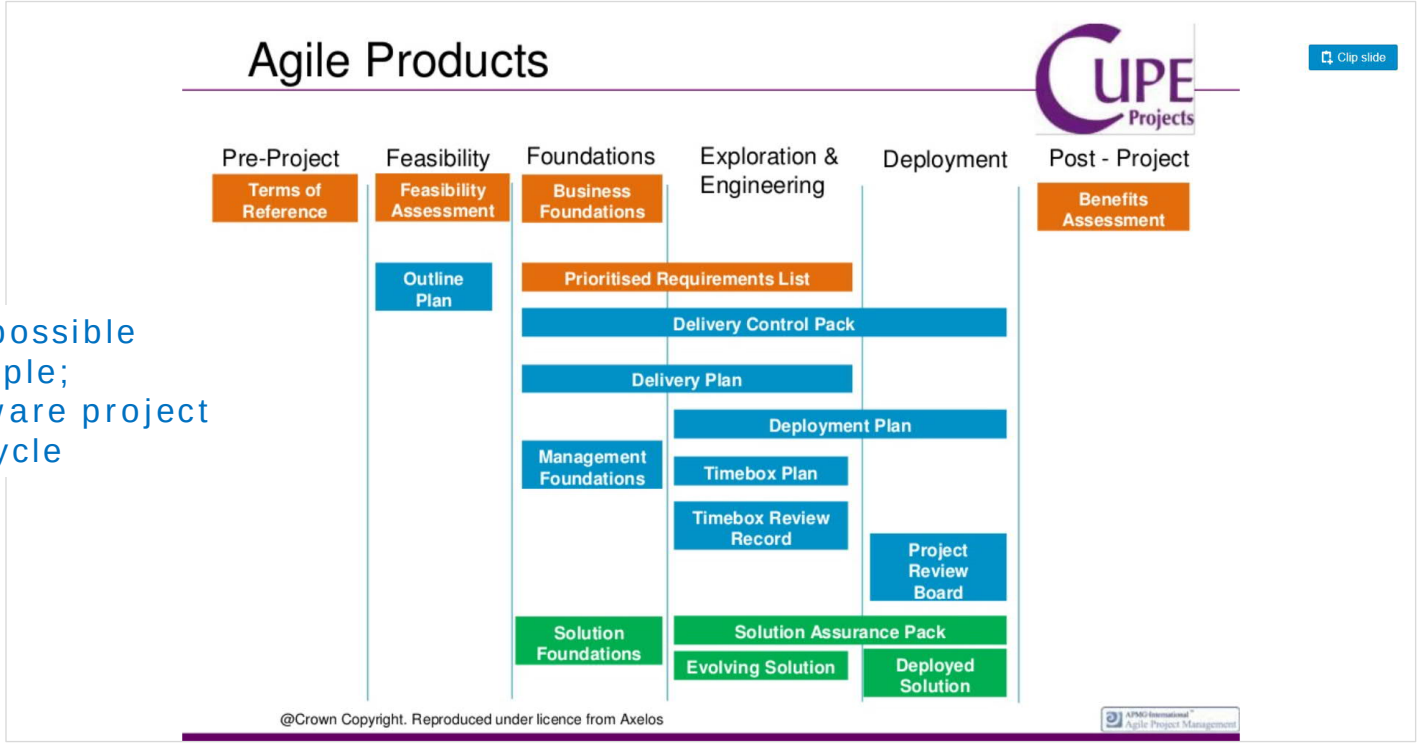
Chris Porter, CTO and co-founder of 6point6 and one of the authors of the report said: “Agile IT in the UK is facing a hidden crisis – 12% of agile projects are failing completely.

You may fail, if you use agile ”wrong”.

[https://www.information-age.com/uk-wasting-37-billion-year-failed-agile-it-projects-123466089/]

[https://www.slideshare.net/CUPE_Projects/]

One possible example; Software project life cycle



[https://www.teamgantt.com/blog/raci-chart-definition-tips-and-example]

Example RACI Chart

Project Deliverable (or Activity)	Project Manager	Strategist	Designer	Front End Developer	Back End Developer
Design site map	C	R	A	I	I
Design wireframes	C	A	R	I	I
Create style guide	A	C	R	C	I
Code templates	A	I	C	R	C

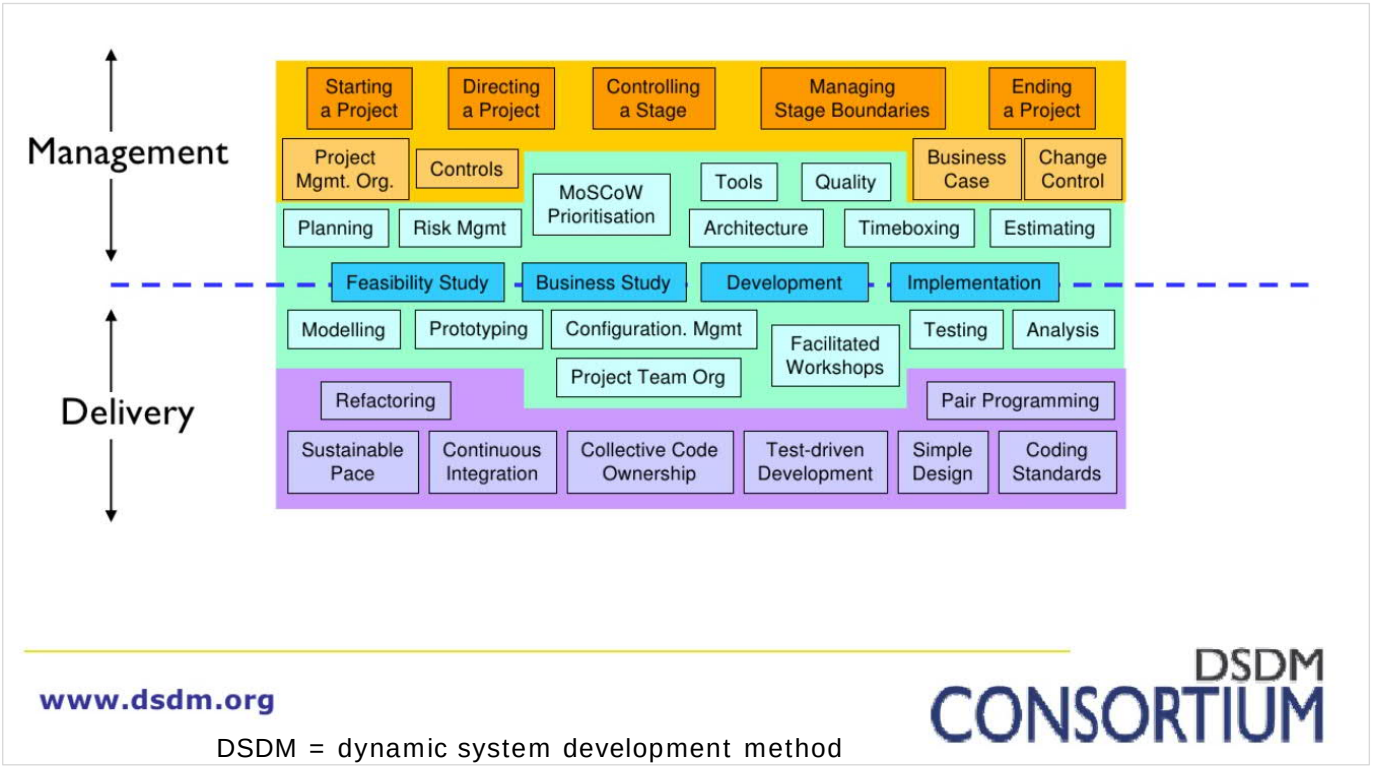
Responsible
The team member who does the work to complete the task

Accountable
The person who delegates work and provides final review on a task or deliverable before it's deemed complete

Consulted
People who provide input on a deliverable based on the impact on their work or their domain of expertise

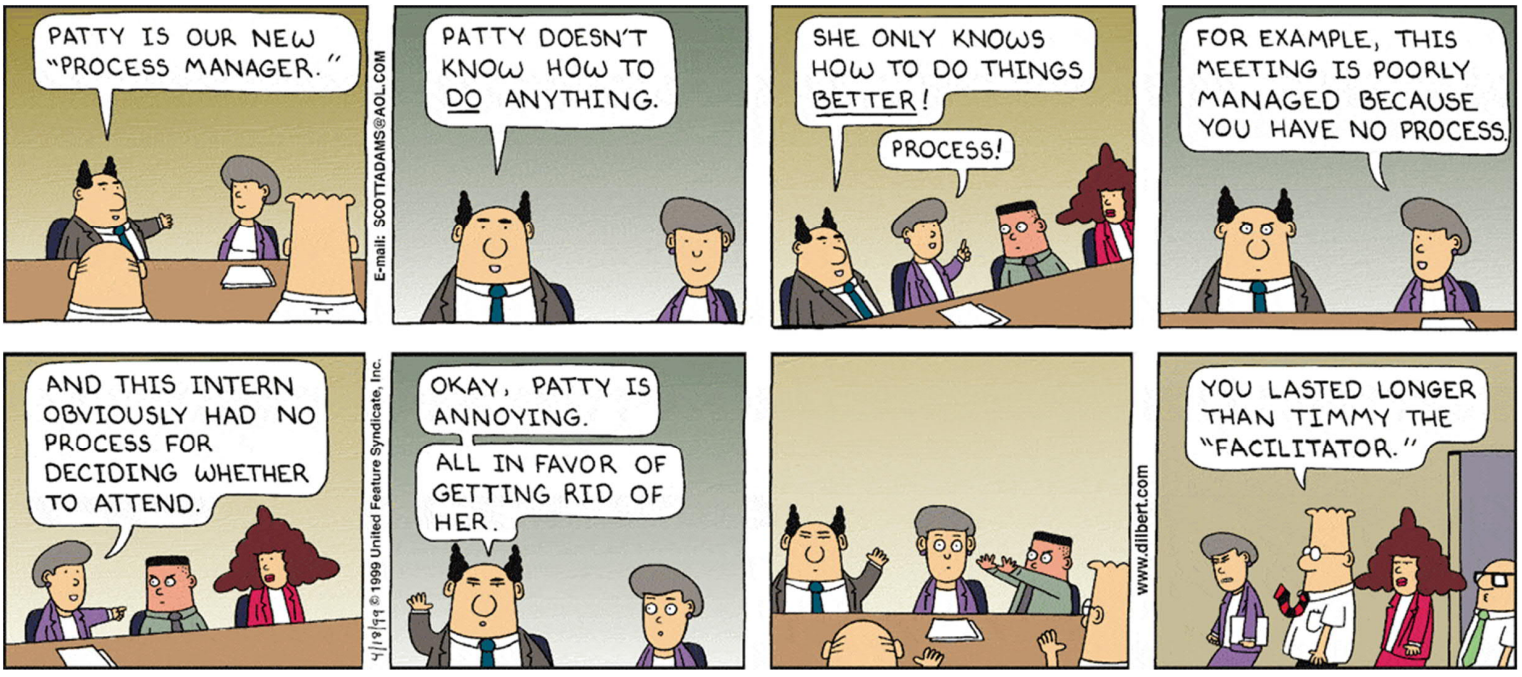
Informed
People who need to be kept in the loop on project progress

Bridging the management – delivery gap

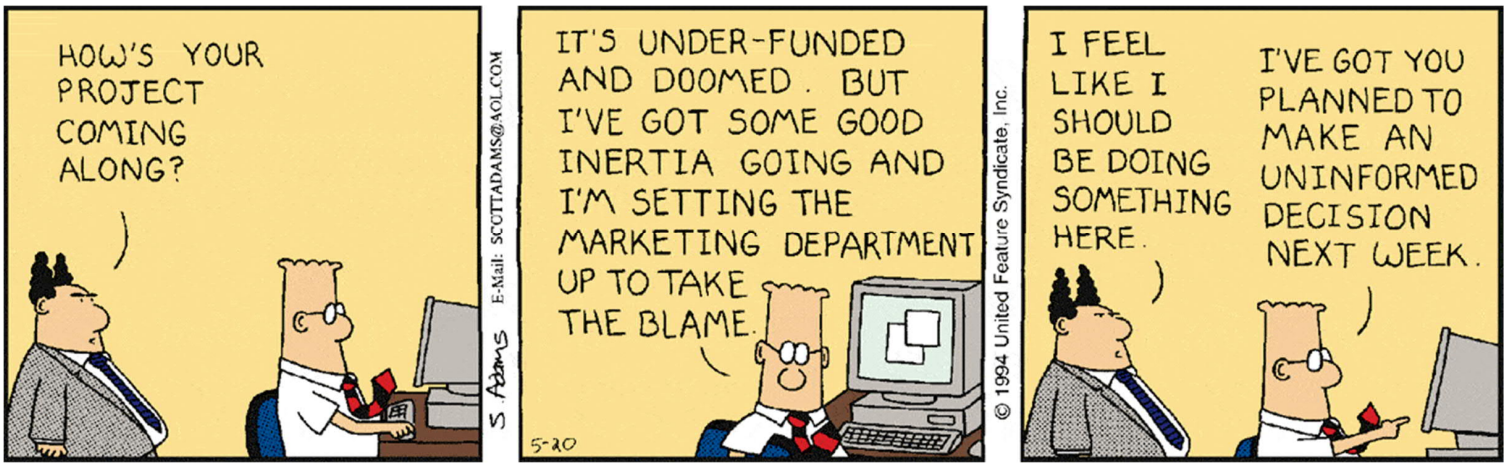


DSDM = dynamic system development method

https://www.projectsmart.co.uk/white-papers.php



Remember only the relevant and essential stakeholders



Tampereen yliopisto
Tampere University

Common success factors

Big Room (FI: allianssimalli)

- An effective Big Room supports cross-functional team collaboration by advancing work and bringing the larger team up to speed on the activities of other groups or individuals. It allows teams to understand their impact across clusters or work groups. The Big Room also provides teams with the time to discuss project-wide concerns like budgets, hot topics, or global changes. The term Big Room refers more to the behaviors and actions of the team than the physical space. The Big Room is more than co-location of people; it is about collaborative behavior and the work they are producing.

[leanconstruction.org]

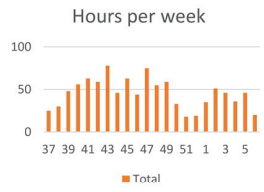
Tampereen yliopisto
Tampere University

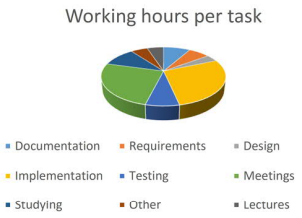
Tools

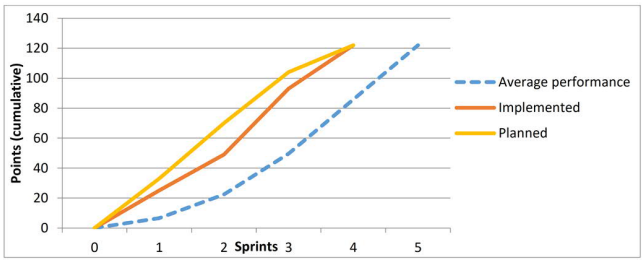
In 396 commits we implemented 140 backlog items with 4 left unimplemented!

We wrote LOC: 16 505 (React might twist this a bit) and we had partial code reuse at around 20%

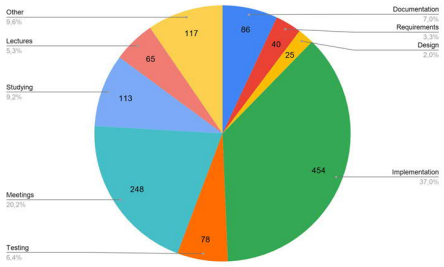
Buzzword bingo: Java, Python, C#, Javascript, Bash, Maven, .NET, React, Flask, JavaFX, Docker, Rest API, git, Semaphore Classic, Slack, Mattermost, GitHub

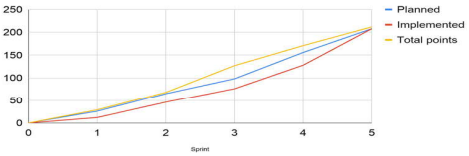






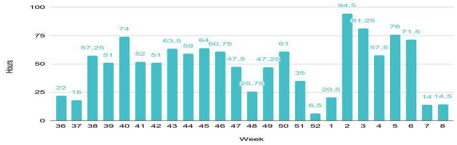
"We've had one, yes. But what about second chaos?"



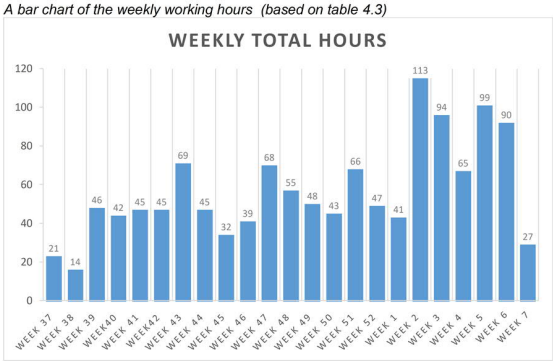


- Tools and technologies used: Typescript, React Native, AWS, Node.js, Express, AWS, Github, Slack, Telegram, TypeORM, Passport.js, PostgreSQL, Expo, CircleCI, Jest, Google Hangouts, ESLint, VSCode, and more.
- **17000** lines of code.
- Modules ("classes"): 43 backend, 42 mobile, 26 web client = **111** total.
- Methods: 505 backend, 251 mobile, 350 web client = **1106** total.
- Commits: 253 backend, 283 mobile, 134 web client = **670** total.
- Issues: 115 implemented, 4 not implemented

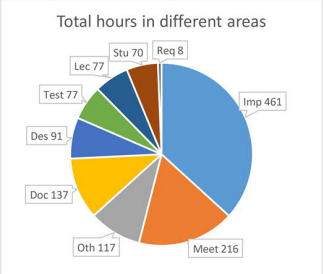
Overall thoughts: Great success!



A bar chart of the weekly working hours (based on table 4.3)



Pie chart of realized working hours / tasks (based on table 4.2)



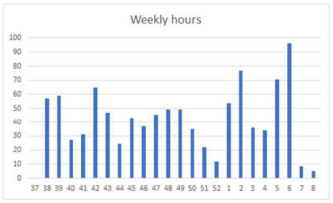
Tools and technologies
Office 365, Android Studio, Java, GitLab, Visual Studio Code, JavaScript, Telegram, SonarQube, React, Material UI, Leaflet and Amazon Web Services: DynamoDB, Lambda, S3, API Gateway.

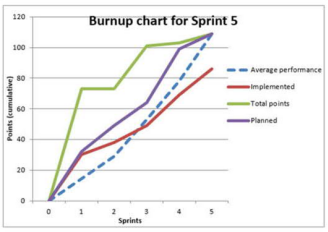
Code statistics

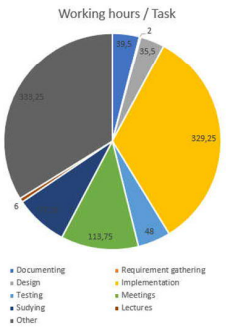
	Frontend	Backend	Android
Lines of code	2808 JavaScript 62 CSS	657 JavaScript 268 YAML	1743 Java 696 XML
Number of classes	6	9	19
Number of methods	236	30	165

Number of revisions: 263 (all components combined)
Implemented items from backlog: 71/81

Hour logs and final burnup chart

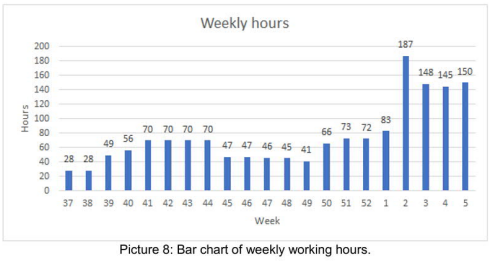




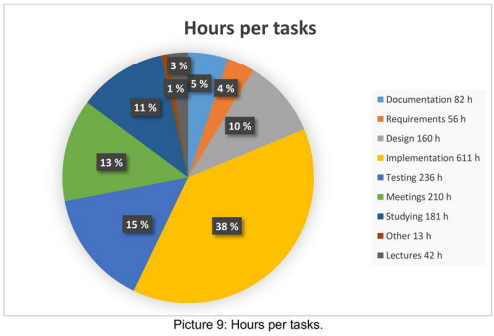


"Voittajana on helppo hymyillä" - "As the winner it's easy to smile."

Tools and technologies: React Native, Node.js/Swagger, PostgreSQL, ASP.NET, GitLab, Heroku SonarQube.



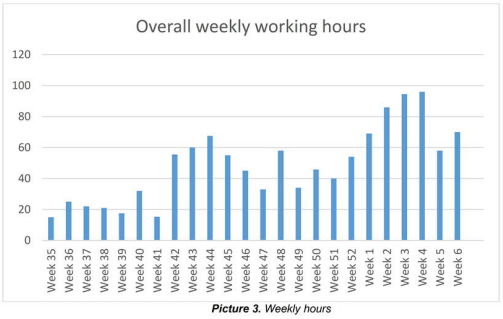
Picture 8: Bar chart of weekly working hours.



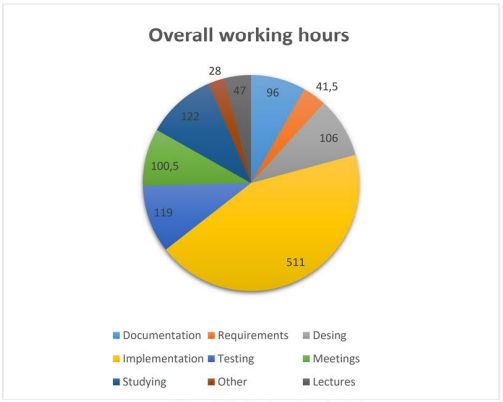
Picture 9: Hours per tasks.

Lines of code: 8500
Reused lines of code: 705
Number of classes: 27
Number of methods: 121
Number of revisions in version control: 304 commits in Git

- Main technologies: Flutter/Dart, NodeJS, Swagger, PostgreSQL, Heroku



Picture 3. *Weekly hours*



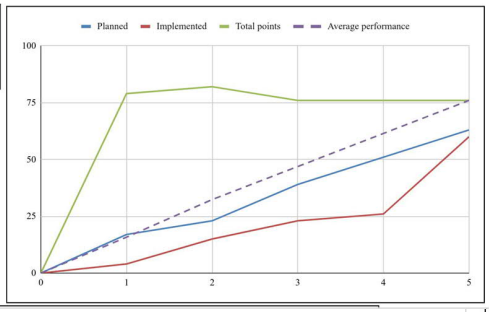
Picture 4. *Working hours based on tasks*

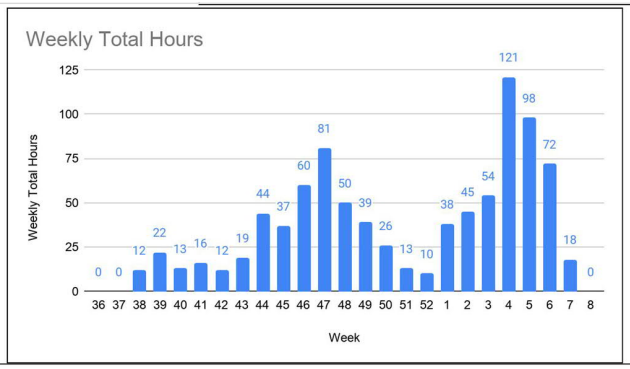
Technologies: C#, ASP.NET Core 3, Angular 8, TypeScript, gRPC, Angular Material, SASS

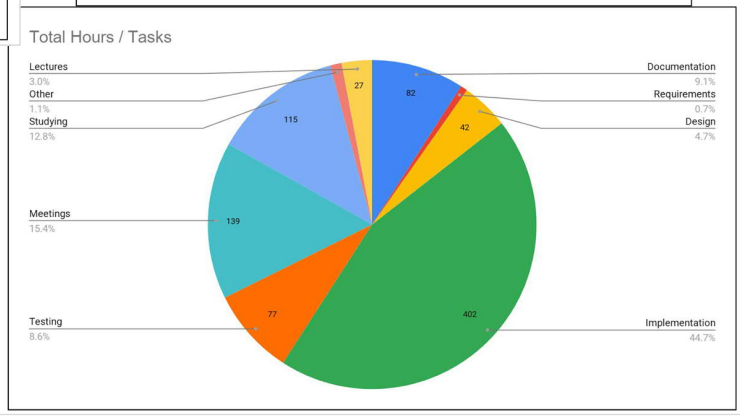
Files	Blank lines	Comment lines	Code lines
242	1622	762	12305

Number of commits in Git: 275
Backlog items: 7 not implemented, 29 implemented, 2 dropped (80.6% done)

Our thoughts: This is fine🔥





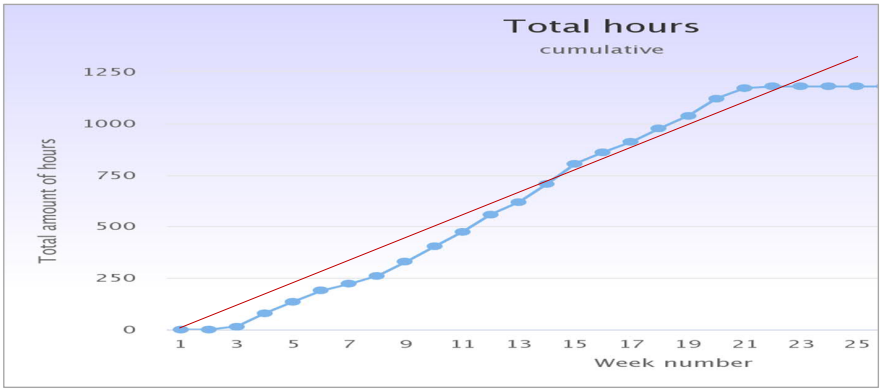


Tampereen yliopisto
Tampere University

TSS 2020 project as an example.

Weeks are real calendar weeks.

”default” amount (10 h / person)



”default” amount

Charts from TUNI MMT (Metrics monitoring tool)

Tampereen yliopisto
Tampere University

Scheduling

Tampereen yliopisto
Tampere University

scheduling, time estimations

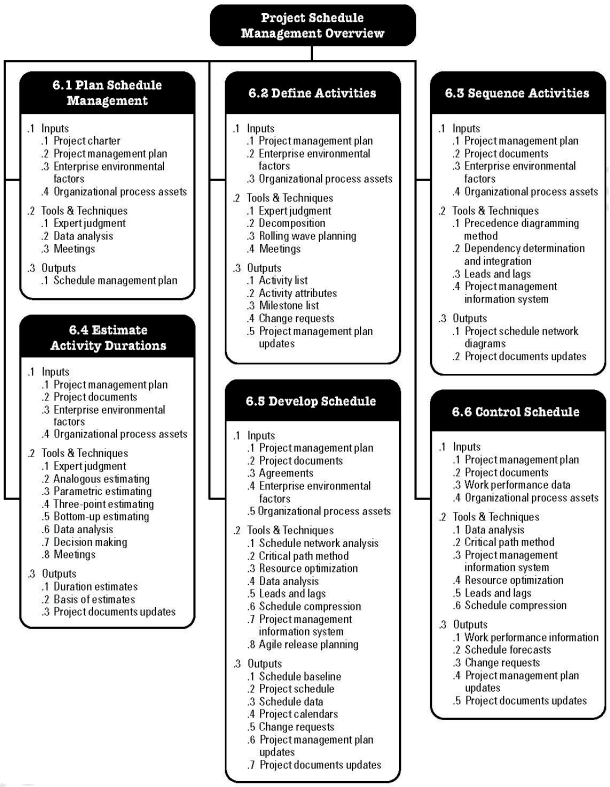
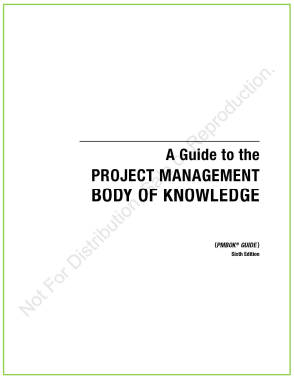


Figure 6-1. Project Schedule Management Overview



[Guide to PMBOK, 2017]

Tampereen yliopisto
Tampere University

scheduling

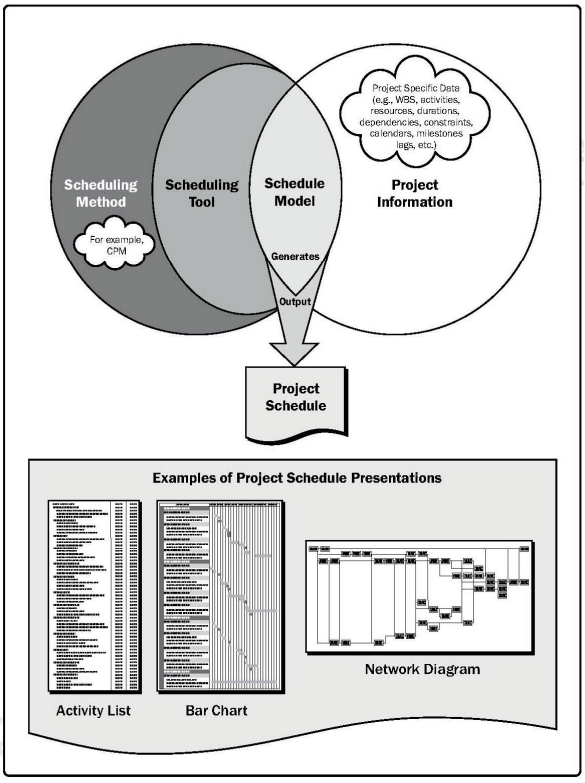


Figure 6-2. Scheduling Overview

[Guide to PMBOK, 2017]

Project scheduling

- Project scheduling is the process of **deciding how the work in a project will be organized** as separate tasks, and **when** and **how** these tasks will be executed.
- You **estimate** the calendar time needed to complete each task, the effort required and who will work on the tasks that have been identified.
- You also have to estimate the resources needed to complete each task, such as the disk space required on a server, the time required on specialized hardware, such as a simulator, and what the travel budget will be.

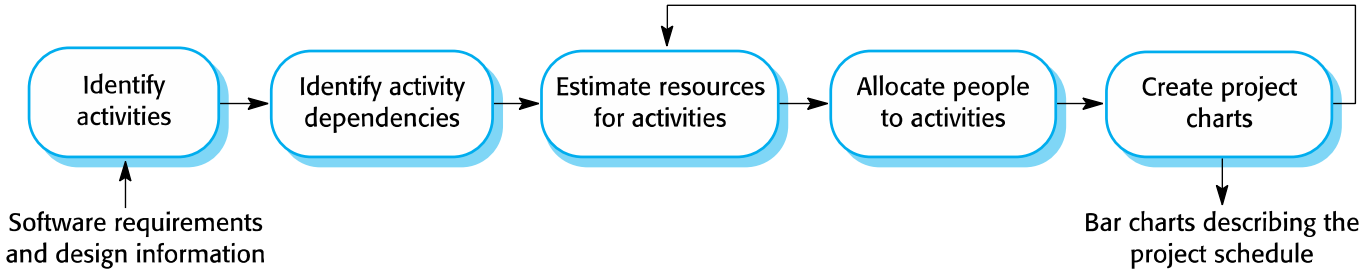
> Whatever you do for the first time, is difficult.
> At second time it goes better and much easier.

Project scheduling activities

- **Split** project into tasks and estimate time and resources required to complete each task.
- **Organize** tasks concurrently to make optimal use of workforce.
- **Minimize** task dependencies to avoid delays caused by one task waiting for another to complete.
- Dependent on project managers intuition and experience.

> You may use top-dow or bottom-up estimation for tasks.

The project scheduling process



Scheduling problems

- Estimating the **difficulty** of problems and hence the **cost** of developing a solution is hard.
- Productivity is not proportional to the number of people working on a task.
- **Adding people to a late project makes it later because of communication overheads**.
- **The unexpected always happens**. Always allow contingency in planning.

Experienced Project Managers spare one weekday at their calendar for unplanned tasks, as a ”buffer” (20 % of weekly working time).

Schedule presentation

- ✧ Graphical notations are normally used to illustrate the project schedule.
- ✧ These show the project breakdown into tasks. Tasks should not be too small. They should take about a week or two.
- ✧ Calendar-based
  - Bar charts are the most commonly used representation for project schedules. They show the schedule as activities or resources against time.
- ✧ Activity networks
  - Show task dependencies

WBS = Work Breakdown Structure, split work to manageable small parts.

Tampereen yliopisto
Tampere University

Metrics

software metrics [Tutorialspoint]

Software metrics can be classified into three categories –

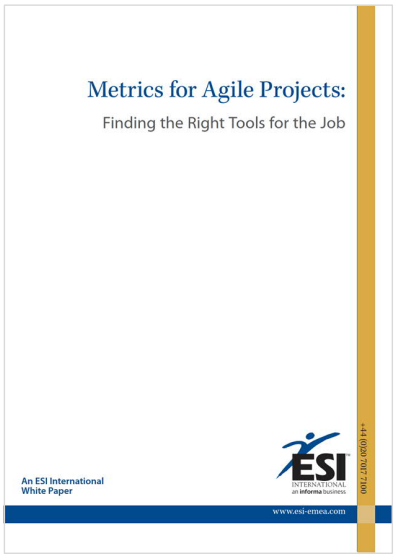
Product metrics – Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.

Process metrics – These characteristics can be used to improve the development and maintenance activities of the software.

Project metrics – This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

If you want to know how a project is going, you have to measure something.
Select the metrics wisely (measurable exact values) and reasonably (which matters).

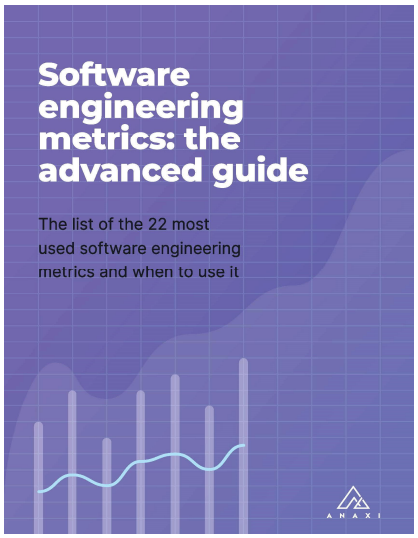
Perhaps it is not reasonable to measure computer processor load, nor worker's blood pressure.



AgileEVM Metric	Definition
PSP (planned story points)	Total number of story points planned for this release
PW (planned weeks)	Total number of planned development weeks
BAC (budget at completion)	Total budget for the release
AW (actual weeks)	Number of development weeks elapsed to date
CSP (completed story points)	Number of story points completed to date
PPC (planned percent complete)	AW/PW
APC (actual percent complete)	CSP/PSP
AC (actual cost)	Total budget spent to date
PV (planned value)	Budgeted cost of the story points that were scheduled to be completed as of today; PPC * BAC
EV (earned value)	Budgeted cost for the story points actually completed as of today; APC * BAC
CV (cost variance)	Difference between planned budget and actual spend; EV – AC
SV (schedule variance)	Difference between planned schedule and actual time spent; EV – PV
CPI (cost performance index)	EV/AC
SPI (schedule performance index)	EV/PV
ETC (estimate to complete)	Based on current state, how much additional budget is needed to complete the release; 1/CPI * (BAC – EV)
EAC (estimate at complete)	Based on current state, what the total estimated cost of the release will be at completion; EAC = AC + ETC
Estimated time to complete	Based on the current state, the total estimated time needed to complete the release; 1/SPI * PW

Exhibit 3: Agile Earned Value Management

JAC (just another classification) of metrics



What are software engineering metrics?

In software, there are 2 categories of metrics and we use different names for those:

1. Software engineering metrics, also known as software development metrics, or software delivery performance, every team has a different name for them, it seems. What is important here is that those indicators measure how software is being built and the engineering team productivity.

2. Software or application performance metrics are the metrics of the software delivered, response time of the application, etc. NewRelic is typically one of the main providers of such metrics. You could also think of this in terms of customer satisfaction metrics - Net Promoter Score typically).

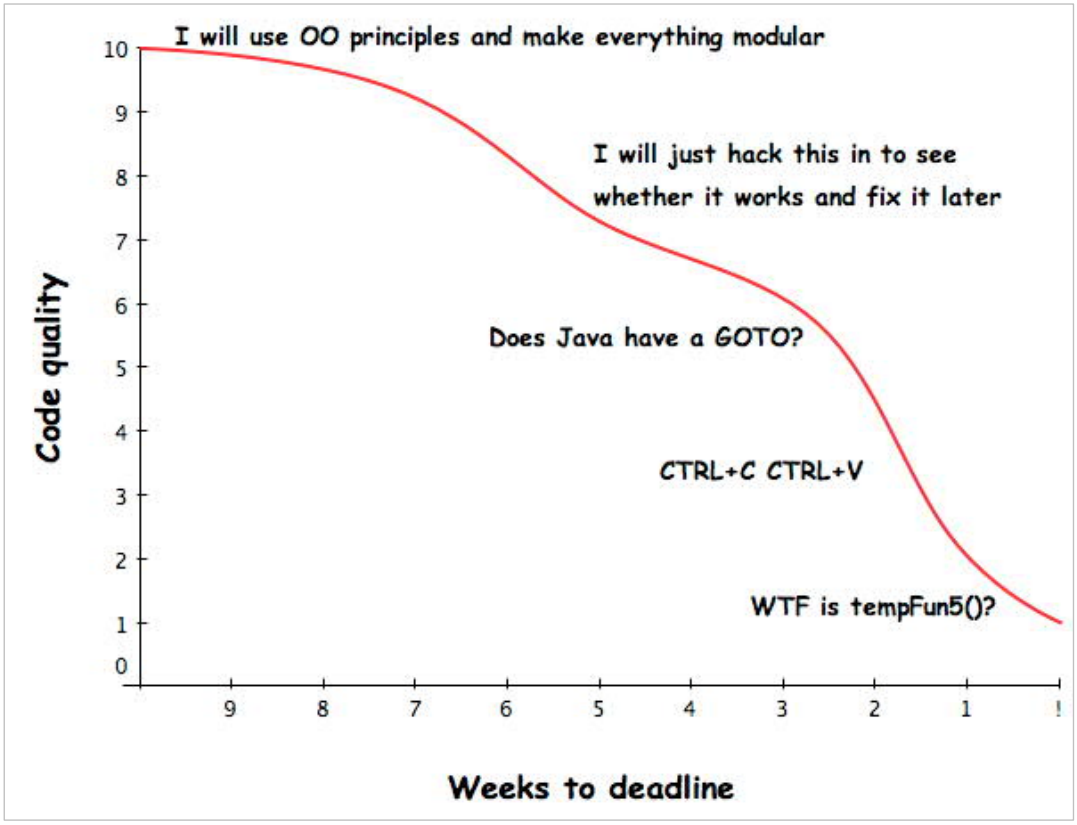
[https://anaxi.com/software-engineering-metrics-an-advanced-guide/]

Top 5 Software Metrics to Manage Development Projects

Types of Software Metrics

- Formal code metrics—Such as Lines of Code (LOC), code complexity, Instruction Path Length, etc. In modern development environments, these are considered less useful.
- Developer productivity metrics—Such as active days, assignment scope, efficiency and code churn. These metrics can help you understand how much time and work developers are investing in a software project.
- Agile process metrics—Such as lead time, cycle time and velocity. They measure the progress of a dev team in producing working, shipping-quality software features.
- Operational metrics—Such as Mean Time Between Failures (MTBF) and Mean Time to Recover (MTTR). This checks how software is running in production and how effective operations staff are at maintaining it.
- Test metrics—Such as code coverage, percent of automated tests, and defects in production. This measures how comprehensively a system is tested, which should be correlated with software quality.
- EXTRA: Customer satisfaction—Such as Net Promoter Score (NPS), Customer Effort Score (CES) and Customer Satisfaction Score (CSAT). The ultimate measurement of how customers experience the software and their interaction with the software vendor.

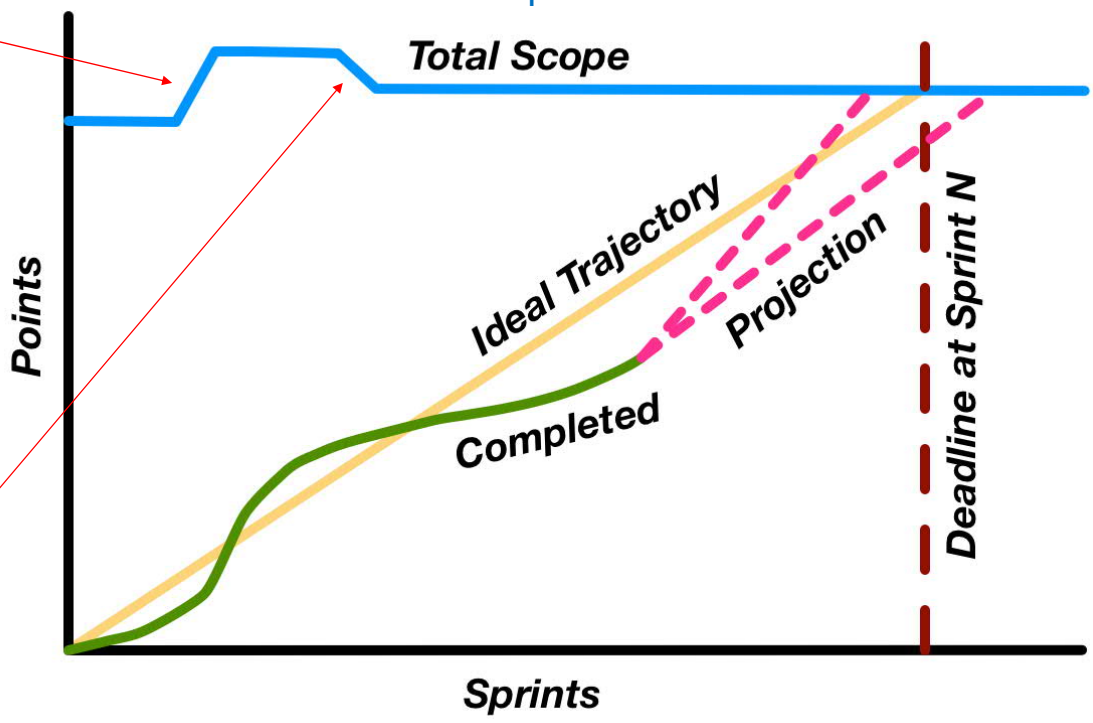
[https://www.sealights.io/software-development-metrics/top-5-software-metrics-to-manage-development-projects-effectively/]



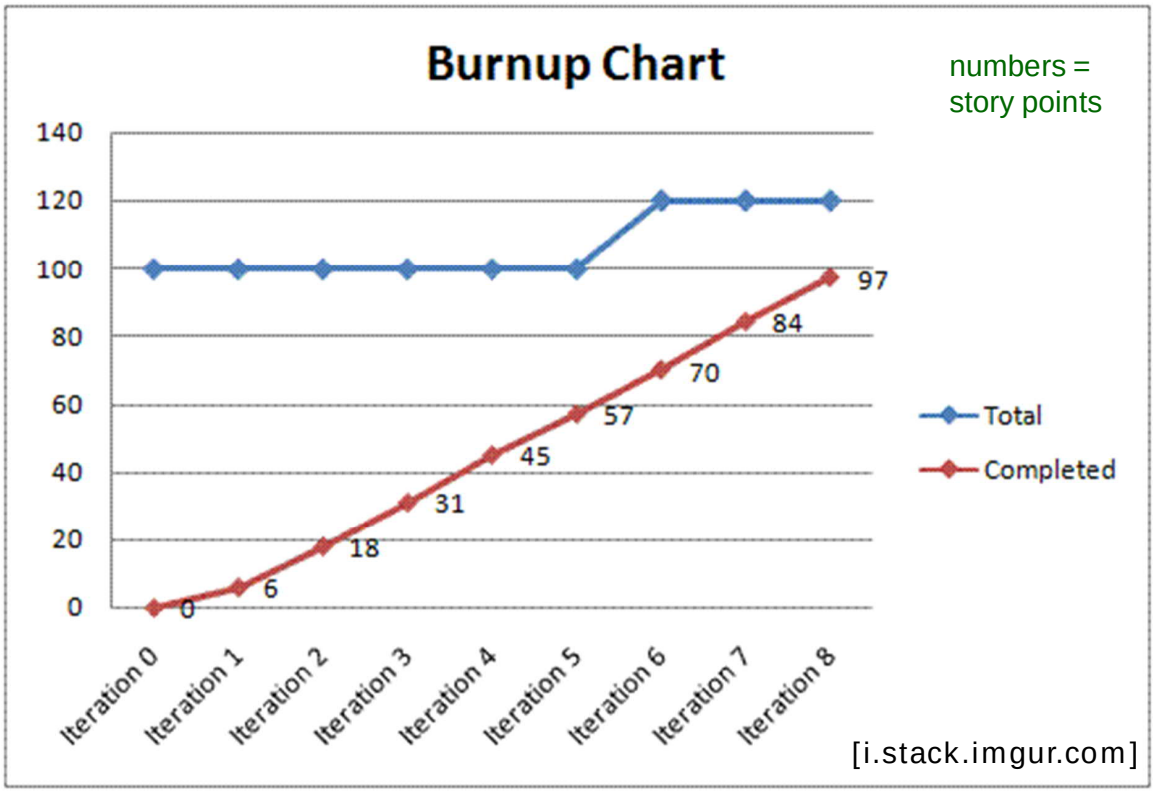
BurnUp chart

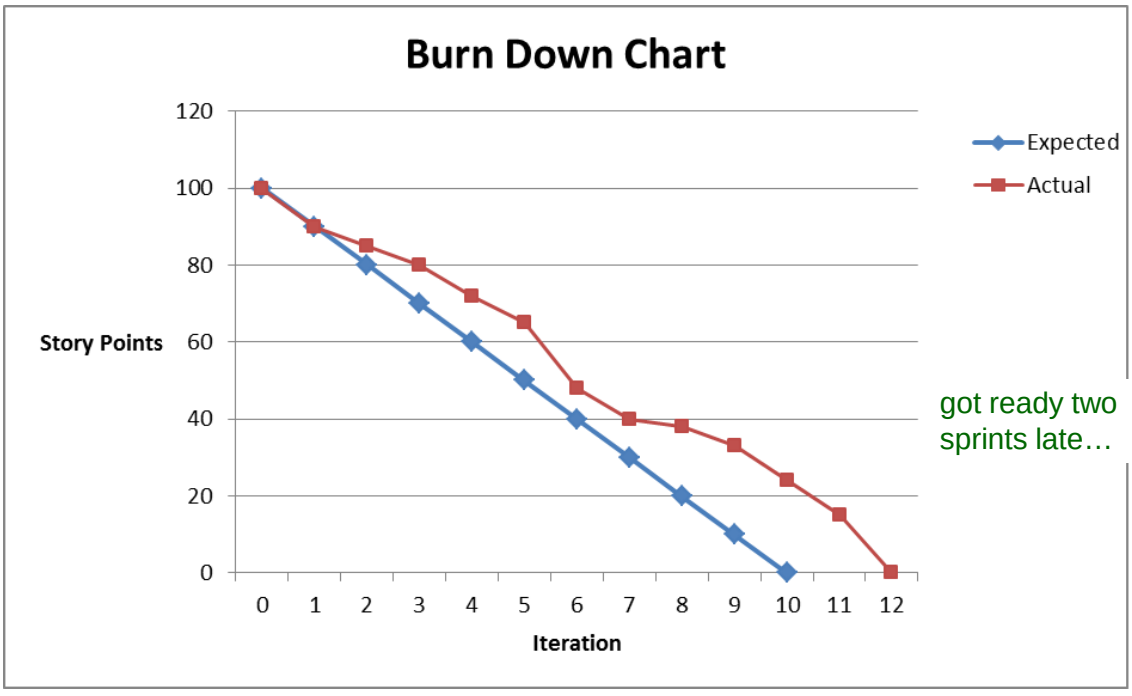
Some more items added to Product Backlog

Some items taken off from Product Backlog



[spin.atomicobject.com]


Burnup Chart
numbers = story points
140
120
100
80
60
40
20
0
0
6
18
31
45
57
70
84
97
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Total
Completed
[i.stack.imgur.com]


Burn Down Chart
Story Points
120
100
80
60
40
20
0
0 1 2 3 4 5 6 7 8 9 10 11 12
Iteration
Expected
Actual
got ready two sprints late…
[blog.caplin.com]

Tampereen yliopisto
Tampere University

FiSMA

ICS › 35 › 35.080

ISO/IEC 29881:2010

Information technology — Systems and software engineering — FiSMA 1.1 functional size measurement method

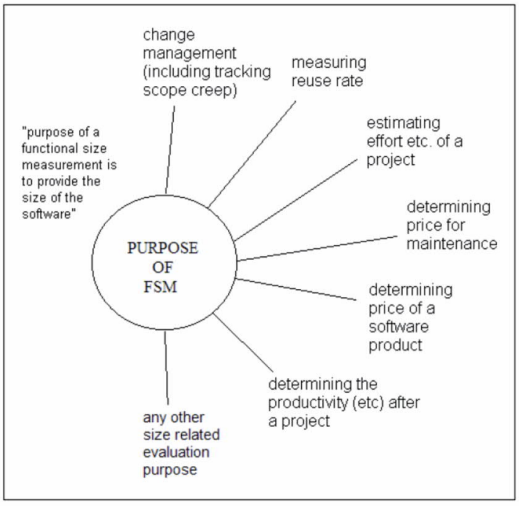


Figure 2 — Common purposes of Functional Size Measurement

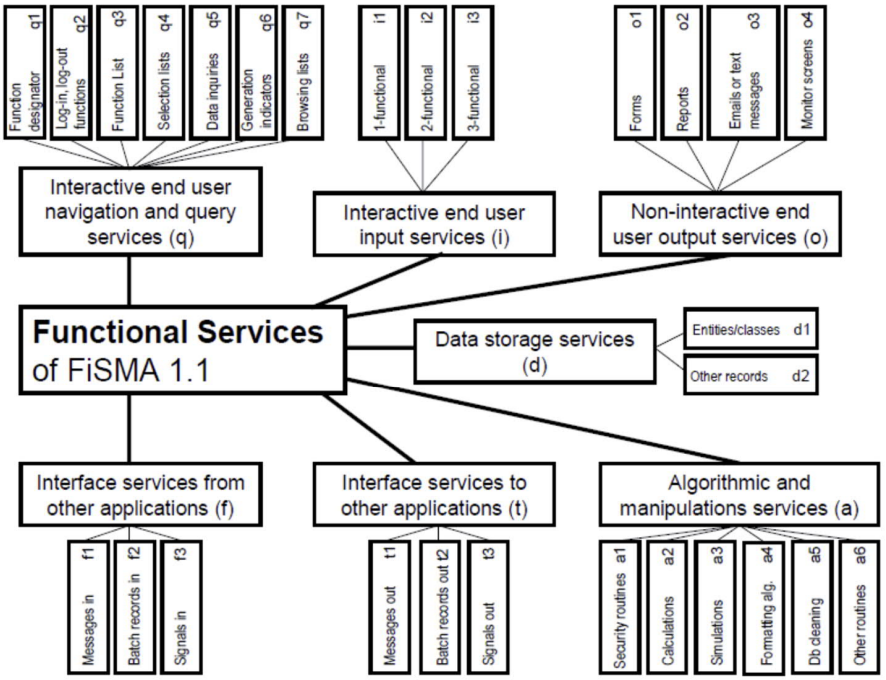
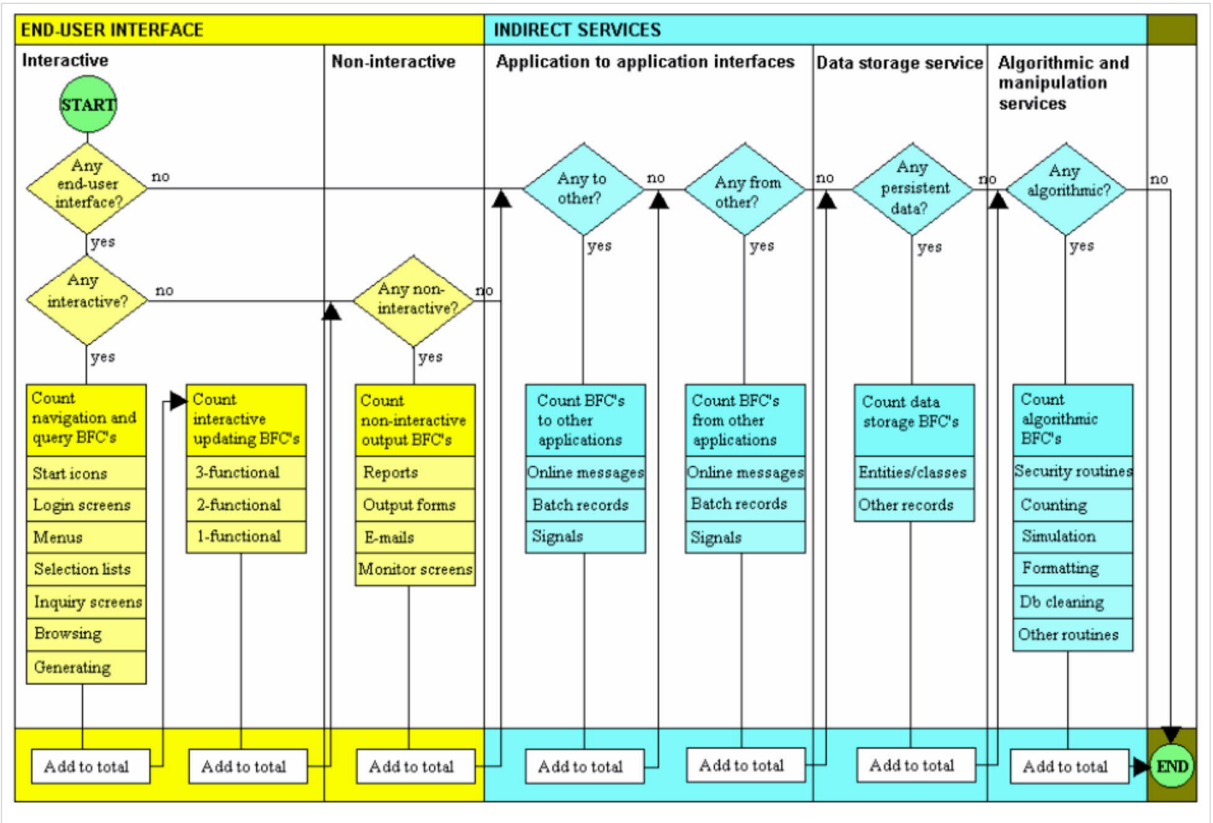


Figure 3 — FiSMA 1.1 BFC classes and BFC types

BFC = base functional component

Tampereen yliopisto
Tampere University

FiSMA process



Finnish Software Measurement Association

FiSMA

Condition based classification

- Typical areas of software metrics are:
  - Software product (itself)
  - The process used to produce software
  - The management of developing software
  - Leading and managing software business
- Other classifiers:
  - The software product standard: internal, external, in use
  - The software process standard: capability, maturity
  - Software supplier / customer organisation / end-user
  - Project model: cost, time, quality, resources, workload, benefit/profit
  - Critical systems: stability, integrity, method conformance
  - Life cycle model: specification, technical planning, development, verification, validation, production (in all e.g. V&V findings, coverage, traceability)
  - Maturity wise: initial, managed, performed sets

FiSMA 2012 5

http://www.4sumpartners.com/world-record-function-point-counting/

WORLD RECORD FUNCTION POINT COUNTING

'Function point counting', i.e. the process of measuring functional size of software, has claimed to be difficult and time consuming. Several experiments, including the ones organized by FiSMA and introduced below, have proven that the claim is not true, if the measurer knows his method, has fair tools and well-documented information available. Regrettably often system documentation (Product Requirements Document) is fuzzy and incomplete so that finding the functionality there is almost impossible, or at least extremely laborious. In the cases below we had organized the experiment so that the documentation was fairly well done and made it easy to find all the software functions required for counting. This was the way to recognize the true speed of size measurement process, without any disturbance from external sources, like exploring through poor documentation or explaining the details of measurement method to software developers and users.

23 Finnish software experts were invited to FiSMA office in Espoo in November 2015. The purpose of the meeting was to try, how fast they can measure functional size of software, if the circumstances are as good as they can. They were given the information of five systems or pieces of software in terms of numbers of different kind of functional components. For comparison, a similar type of experiment was arranged also 18 months earlier for a larger group of members of the FiSMA Scope Manager Forum. In that case they had measured software size using a professional tool, Experience® Service. The old "world record" from that case was measuring five applications with average speed of 2.7 minutes per software. The average measurement error of 60 measurers was only 1 % compared to the correct result. There they also completed effort estimation in a few extra minutes, still with astonishing success rate. This experiment has documented and published in a paper of MetriKon conference.

How to estimate with COCOMO II

1. Estimate scaling factor SF_i (I=1..5) and calculate scaling exponent E.
2. Split product to independent parts to be estimate.
3. Estimate size of each part, considering reuse and automatic code generation.
4. Estimate cost factors EM_i and calculate nominal person months PM_{ns}.
5. Calculate development time $TDEV_{NS}$.
6. If one want to deliver in shorter time. Choose SCED value "low" or "very low", when TDEV drops max 25%. Correspondingly PM increases due to bigger SCED-factor.

COCOMO (I - 1981, II - 2000)

COnstructive COst MOdel

Nominal effort (person-months)

- $PM_{NS} = A * Size^E * EM_1 * EM_2 \ldots * Em_n$

A = effort coefficient, that can be calibrated according to situation

- Based on large data set Boehm proposes 2.94
- Could be interpreted as months per KSLOC

Size is KSLOC

- EM_i are cofficients (7 in early design and 17 in post design)

E is a scaling exponent between 1.1 and 1.24 – depending on novelty

- $E = 0.91 + 0.01 *(SF_1 + SF_2 + SF_3 + SF_4 + SF_5)$.

COCOMO

Development time, calendar time

$TDEV_{NS} = C * (PM_{NS})^F$

C is calibration factor, some material gives 3.67

F is a scaling exponent,

– $F = D + 0.2 * 0.01 * SF_1 * SF_2 * SF_3 * SF_4 * SF_5$

$= D + 0.2 * (E - 0.91)$

Where

D is a calibration term, some material propose 0.28.

Table 8: SRM Defect Prediction for Waterfall Development

1000 function points; inexperienced team; CMMI Level 1; Waterfall; C language
106,670 logical code statements; 106.7 KLOC

Defect Potentials	Defects	Per FP	Per KLOC	Pre-Test Removal	Test Removal	Defects Delivered	Cumul. Effic.
Requirements	1,065	1.07	9.99	70.00%	54.00%	147	86.20%
Design	1,426	1.43	13.37	76.00%	69.00%	106	92.56%
Code	1,515	1.52	14.20	77.00%	74.00%	91	94.02%
Documents	665	0.66	6.23	79.00%	19.00%	113	82.99%
Bad fixes	352	0.35	3.30	59.00%	61.00%	56	84.01%
TOTAL	5,023	5.02	47.09	74.24%	60.36%	513	89.79 %

SRM = Software Risk Master tool. [Capers Jones, 2013]

Some other standards

- ISO/IEC 14143-1:2007

Information technology -- Software measurement -- Functional size measurement -- Part 1: Definition of concepts

ISO/IEC 14143-1:2007 defines the concepts of FSM (Functional Size Measurement). The concepts of Functional Size Measurement (FSM) are designed to overcome the limitations of earlier methods of sizing software by shifting the focus away from measuring how the software is implemented to measuring size in terms of the functions required by the user.

- ISO/IEC 20926:2009

Software and systems engineering -- Software measurement – IFPUG functional size measurement method 2009

ISO/IEC 20926:2009 specifies the set of definitions, rules and steps for applying the IFPUG (International Function Point Users Group) functional size measurement (FSM) method.

Function Points, Story Points, and Velocity on Agile Projects

- Function point metrics are not widely used on Agile projects, in part because manual counting of function points is too slow and expensive to fit the Agile philosophy. Many agile projects use story points as an alternate metric, and they use velocity as a tool for predicting completion of stories in a specific time period such as a week or a month.
- The high-speed sizing method of Software Risk Master ™ (SRM) which sizes applications in about 1,8 minutes is a good match to the agile philosophy. To facilitate use by agile projects SRM can also do bi-directional conversion between story points and use case points. It can also predict velocity.

[Capers Jones 2013]

Some metrics books...

- IFPUG: IT Measurement – Practical Advice from the Experts. 2002, Addison-Wesley. 759 p.
- Fenton and Pfleeger: Software Metrics, rev. Print. 1997, PWS Publishing company. 638 p.
- COCOMO II - Model Definition Manual, 2000
- IFPUG: Function Point Counting Practices Manual, Release 4.1.1, 2000
- Function Points Analysis Training Course

”Overkill” manuals:

- NASA Cost Estimating Handbook, 2008
- Software Cost Estimation Metrics Manual for Defense Systems, 2015.

Function points

EIF = external interface file
EQ = external inquiry
EI = external input
EO = external output

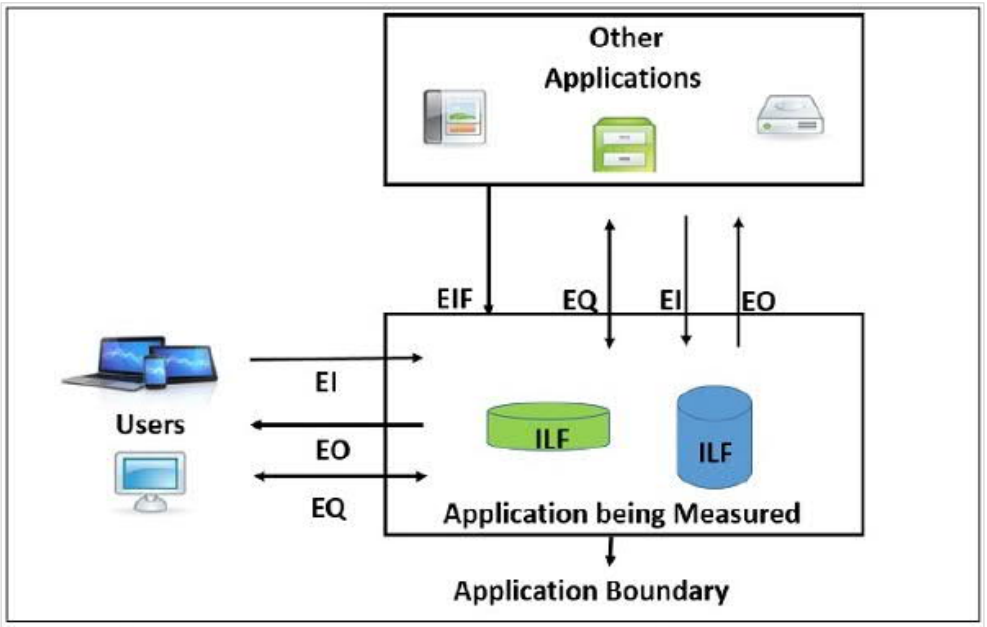


Figure 1: Application Boundary, Data Functions, Transaction Functions

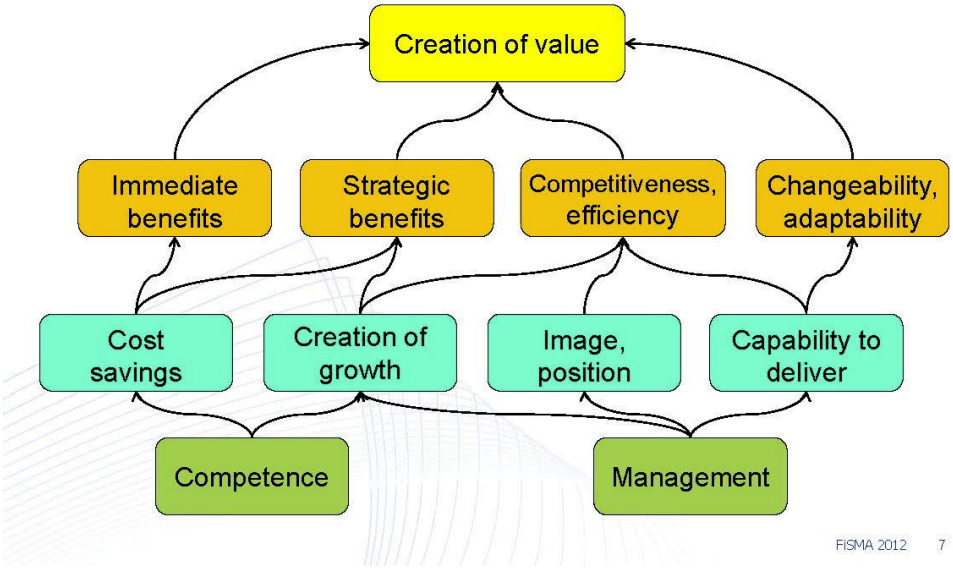
[www.tutorialspoint.com/estimation_techniques/]

FiSMA = Finnish Software Measurement Association

Finnish Software Measurement Association

FiSMA

Measurement needs



FiSMA 2012 7

Actually in Finland many many years ago one company had this kind of ”bug bonus”, but just for a few months.

FiSMA 1.1

Calculate these three different elements/references from the E-Store SRS:

- data element = a character string, or a digital or graphical element
- reading reference = data storage entity or record, from another software or system
- writing reference = data storage entity or record, to another software or system.

Basically, every single requirement/functionality is at least one Ffp, many will be worth more Ffps.

Ffp = FiSMA function point.

FiSMA 1.1, chapter 6

The general counting rules of 6.1..6.7 is

$$S = a + n/d + r/c + w/b$$

- S = functional size
- a = constant
- n = number of data elements, fields (from SRS)
- d = BFC num. of data elements
- r = number of reading references (from SRS)
- c = BFC num. of reading data elements
- w = number of writing references (from SRS)
- b = BFC num. of writing data elements.

BFC: d = avg. 5,4 (4,00..7,00), c = avg. 2,00 (1,50..3,00), a = 0,53 (0,1..1,5), b = 1,00.

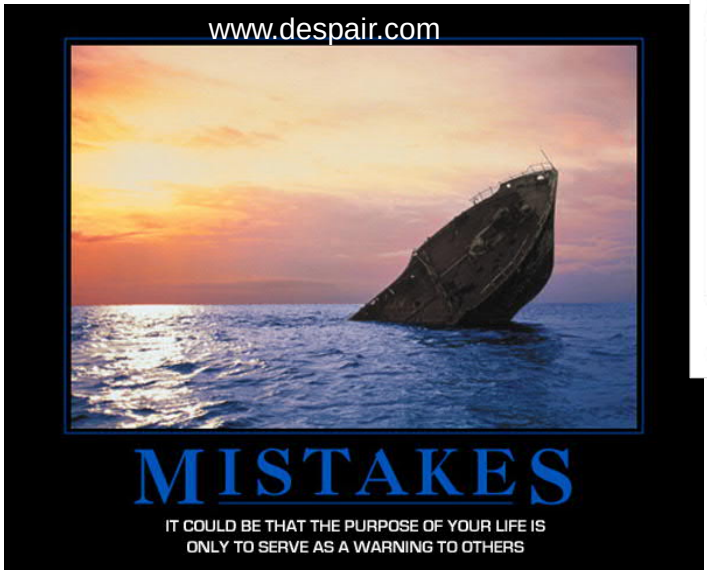
BFC = ("BFC class specific number of …"), constant values are given for these.

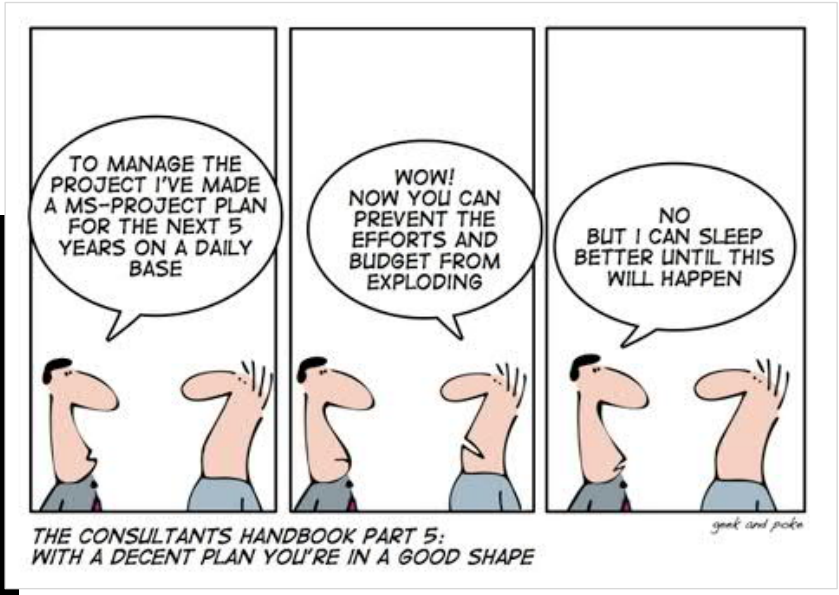
Tampereen yliopisto
Tampere University

Indicators

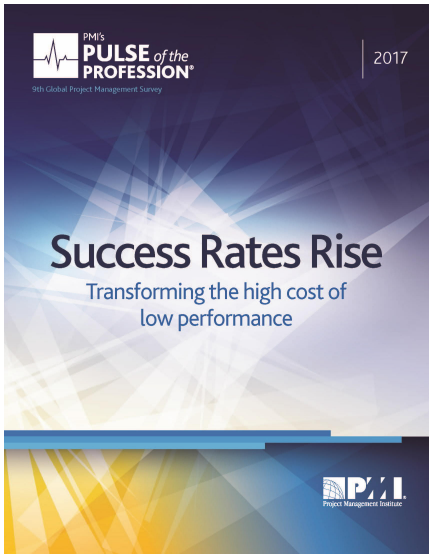
Tampereen yliopisto
Tampere University

Change management



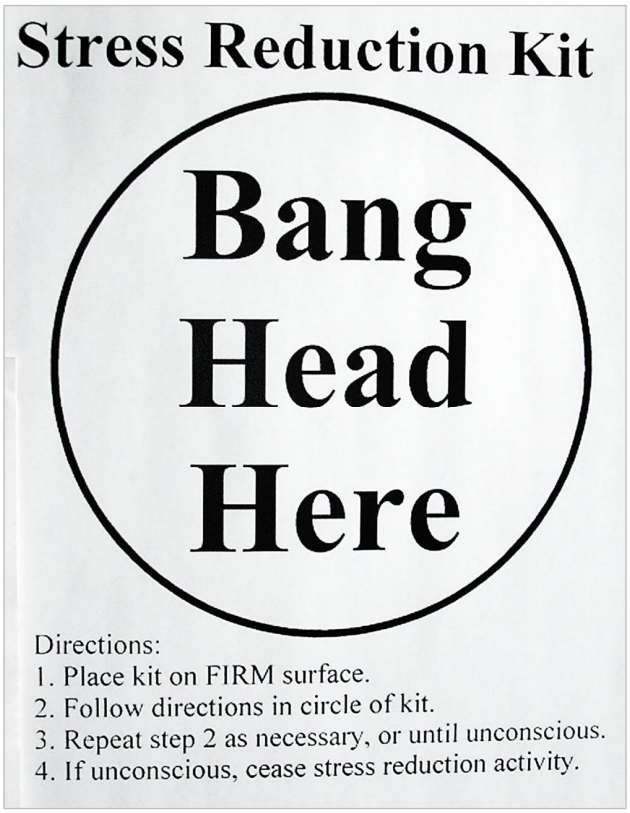


"How do you want your software ?
FAST - CHEAP – GOOD, pick two."



[hackernoon.com]

This SRK was made mainly for project managers, but it works also for other project personnel and stakeholders.



Sometimes project termination or cancellation would be a good option.

Tampereen yliopisto
Tampere University

Risk management

Risk management plan

Risk definition	Trigger (how we get know that risk case came true)	Probability	Impact	Value (P*I)	Main strategy	Alternate strategy (optional)	Action plan for main strategy	Action plan for secondary strategy
Not enough of server power	**Request delay is higher than 1 sec for:** - Top predictors screen (List of user's); - Predict match (Matches schedule, teams table etc.); - My predictions screen (List of predictions).	2	3	6	Transfer	Accept	Reserve additional budget to improve server configuration	We can accept delay, if it is not higher than 2 seconds and we have no complaints from users.
Fixtures API send to us wrong information	**Someone can find out mistakes in app data**	5	4	20	Mitigate	-	"To mitigate that risks we need to add to backlog and develop a new user story. As an Admin I can manage fixtures in admin panel. Affects: additional sprint (scope, budget and timeline)"	-
Fixtures API doesn't send to us information	**We don't get any information** - cup schedule is empty (update after match); - match results (during and after match) is empty;	5	5	25	Mitigate	-		
PlayMarket or / and AppStore release issues (general)	**In case of** - AppStore / PlayMarket can reject an app publishing; - AppStore / PlayMarket can review an app too long.	3	3	9	Mitigate	-	"Reserve additional 1-24h for possible changes Affects: additional 1-3 day (budget and timeline)"	-
PlayMarket or / and AppStore release issues (gambling)	In case of App Store / PlayMarket can reject an app publishing for gambling suspicion	3	10	30	Mitigate	-	"Reserve additional 1-40h for possible changes and appeal Affects: additional 1-30 days (budget and timeline);"	-

[https://themindstudios.com/blog/content/images/2019/12/Risk-management-plan-1.jpg]

EN 16601-80:2014
Space project management - Part 80: Risk management

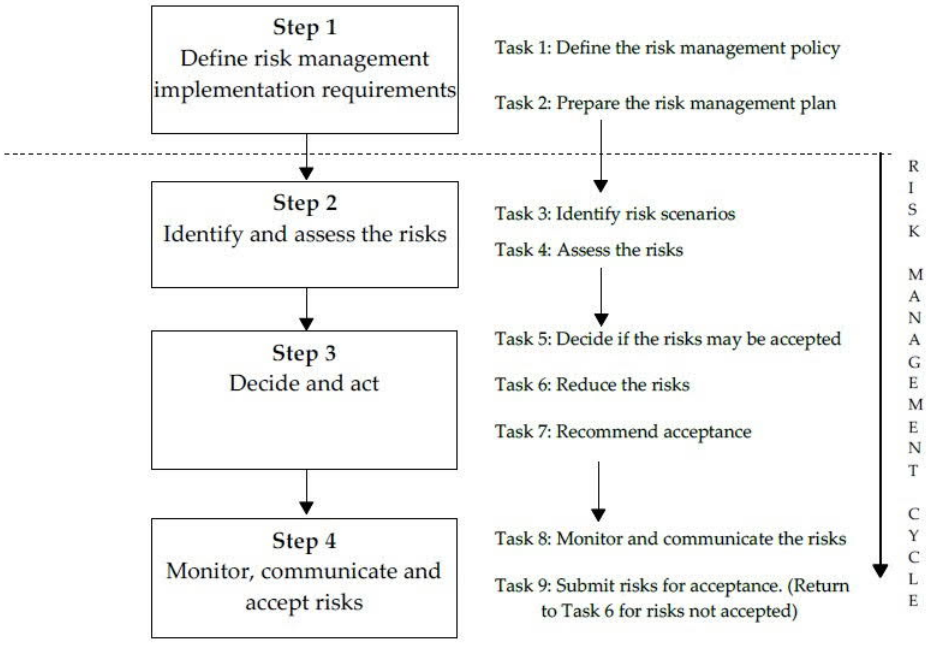


Figure 5-2: The tasks associated with the steps of the risk management process within the risk management cycle

Size-Complexity Matrix

		COMPLEXITY				
		Very Simple	Simple	Average Complexity	Complex	Very Complex
SIZE	Small	100	250	400	550	625
	Moderate	175	325	475	625	775
	Medium	250	400	550	700	850
	Large	325	475	625	775	925
	Grand	400	550	700	850	1000

The Size-Complexity Matrix provides guidelines for categorizing a project in order to assess the risk and effort. The Size-Complexity Matrix uses a 5-point scale for both size and complexity. The lowest-point project is a simple, small project and has 100 points. The largest and most complex project has 1,000 points. Green means low risk and effort, yellow means medium risk and effort, and red means high risk and effort.

[Standish Group, 2019]

Tampereen yliopisto
Tampere University

Size Guidelines		
Size Description		Size
Under $1 million labor	6 or less team members/months	Small
$1 million to $3 million	7 to 12 team members/months	Moderate
$3 million to $6 million	13 to 24 team members/months	Medium
$6 million to $10 million	25 to 50 team members/months	Large
Over $10 Million	Over 50 team members/months	Grand

Guidelines on how to measure the size of a project.

These are the classes for Size-Complexity matrix.

Complexity Guidelines	
Environment	Points
Diverse User Base	1
Multiple Team Locations	1
Multiple Stakeholder Locations	1
Uncooperative Peers	2
Uncooperative Stakeholders	3
Scope	Points
Many Requirements - Large scope	1
Ambiguous Basic scope	1
Fuzzy Undefined Requirements	1
Diverse and Multifaceted Objectives	2
Breaking New Ground	3

Guidelines on how to measure the complexity of a project.

[Standish Group, 2019]

Tampereen yliopisto
Tampere University

Just another risk seriousness matrix

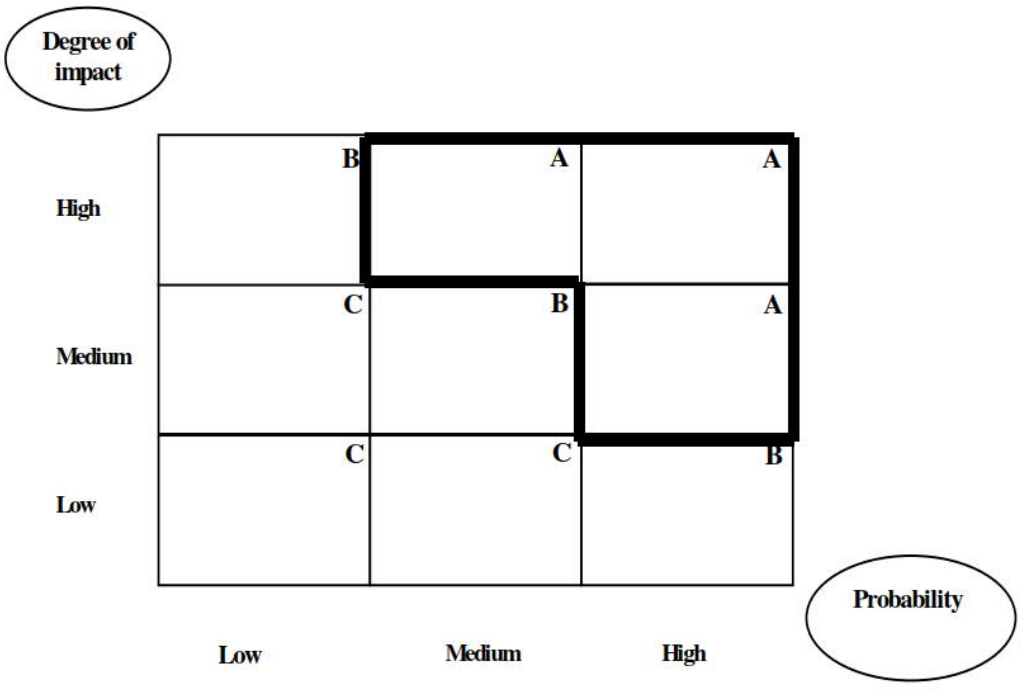


Figure 7: Risk portfolio

[E. WALLMÜLLER: Risk Management for IT and Software Projects]

Tampereen yliopisto
Tampere University

Personal and organisational risk assessment and management,

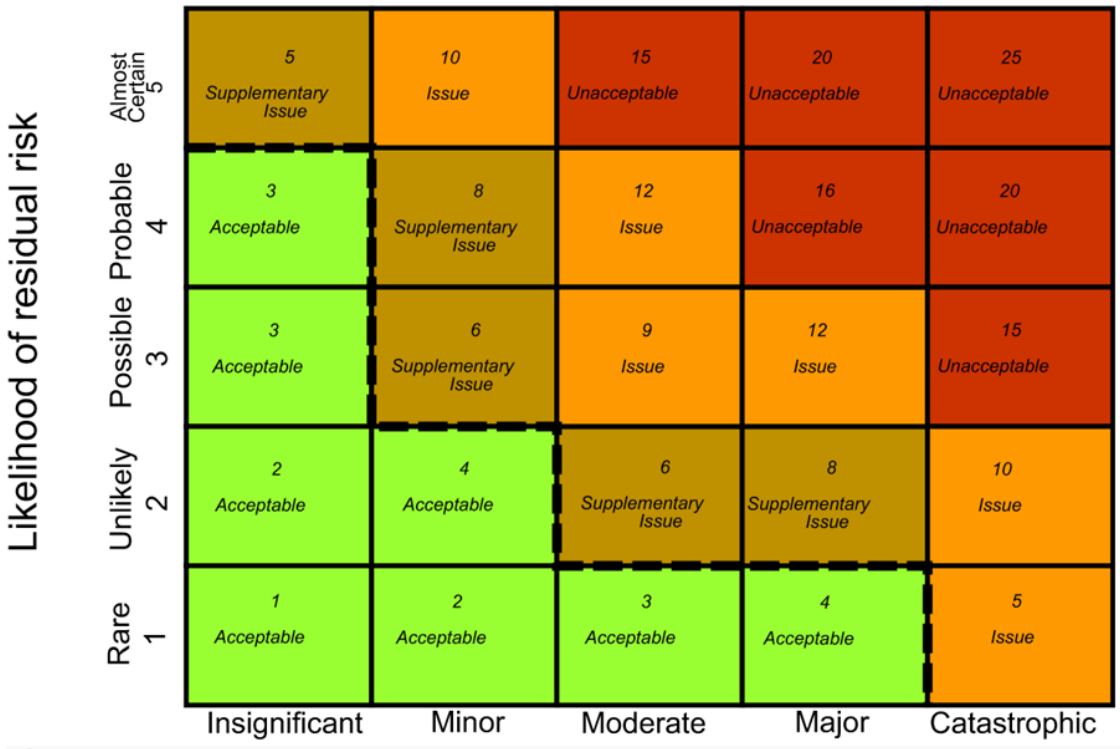
but same kind of 5x5 matrix fits to all kind of risks

[https://blog.codecorps.org/]

		Probability (expected frequency)				
		Frequent: Continuous, regular, or inevitable occurrences	**Likely:** Several or numerous occurrences	**Occasional:** Sporadic or intermittent occurrences	**Seldom:** Infrequent occurrences	**Unlikely:** Possible occurrences but improbable
Severity (expected consequence)		**A**	**B**	**C**	**D**	**E**
Catastrophic: Death, unacceptable loss or damage, mission failure, or unit readiness eliminated	**I**	**EH**	**EH**	**H**	**H**	**M**
Critical: Severe injury, illness, loss, or damage; significantly degraded unit readiness or mission capability	**II**	**EH**	**H**	**H**	**M**	**L**
Moderate: Minor injury, illness, loss, or damage; degraded unit readiness or mission capability	**III**	**H**	**M**	**M**	**L**	**L**
Negligible: Minimal injury, loss, or damage; little or no impact to unit readiness or mission capability	**IV**	**M**	**L**	**L**	**L**	**L**
Legend EH – extremely high risk H – high risk L – low risk M – medium risk						

Tampereen yliopisto
Tampere University

[https://www.guru99.com/risk-analysis-project-management.html]



1..4 = acc
5..8 = supp iss
9..12 = issue
15-25 = unacc

(5=sup iss)

Tampereen yliopisto
Tampere University

PROJ Example risk management list 1

Project Risks

Risk	Impact	Probability	Actions
Customer withdraws	High	Very Low	Edit
Customer doesn't know what they want	Medium	High	Edit
Group members ill/ temporarily unavailable	Medium	High	Edit
Group members quitting	Medium	Very Low	Edit
Lacking technological knowledge	Medium	Low	Edit
Insufficient / bad planning	High	Low	Edit
Communication problems	Low	Low	Edit
Insufficient time	Very High	Medium	Edit
Technical difficulties	Medium	Low	Edit

Project Risks

Risk	Impact	Probability	Actions
Develope environment not avaible	High	Very Low	Edit
Expectations not met	Medium	Medium	Edit
Time managment not sufficient	Very High	High	Edit
Code quality not up to standarts	Low	Low	Edit
Design not suitable for user environment	High	Low	Edit
Required functionality not met	High	Low	Edit
Team member loses ability to work	High	Very Low	Edit
Development machine brokes	Very Low	Very Low	Edit

Tampereen yliopisto
Tampere University

PROJ Example risk management list 2

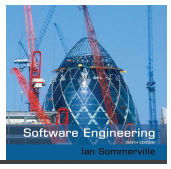
Project Risks

Risk	Impact	Probability	Actions
Lack of communication between team members	Very High	Low	Edit
Lack of communication between team and custom	Very High	Low	Edit
Misunderstanding on requirements	Very High	Low	Edit
Remote working only	Medium	Medium	Edit
Workload goes crazy high near the project end	Very High	High	Edit
Estimated workload has been wrong since start	High	Medium	Edit
Someone does not work as much as promised	High	Medium	Edit
Someone comes too sick to work several weeks	Very High	Very Low	Edit
Busy schedule with school and work	High	Medium	Edit

Project Risks

Risk	Impact	Probability	Actions
Misunderstanding in requirements	Very High	Medium	Edit
Productivity issues	Medium	High	Edit
Too much new to learn in too little time	Medium	High	Edit
Cannot access dev kit due to corona lockdown	High	Medium	Edit
Developers get ill	Medium	Medium	Edit
Developer drop out of the team	High	Low	Edit
Used packages and plugins are bad quality	High	Low	Edit
Focus shift from initial plans	High	Low	Edit
Customer decides to end the project	Very High	Very Low	Edit
Communication problems	Very Low	Medium	Edit

Risk management



- Risk management is concerned with **identifying risks** and drawing up plans to **minimise their effect** on a project.
- Software risk management is important because of the inherent uncertainties in software development.
  - These uncertainties stem from loosely defined requirements, requirements changes due to changes in customer needs, difficulties in estimating the time and resources required for software development, and differences in individual skills.
- You have to anticipate risks, **understand the impact** of these risks on the project, the product and the business, and take steps to **avoid** these risks.

Risk classification

- There are two dimensions of risk classification
  - The **type** of risk (technical, organizational, ..)
  - what is **affected** by the risk:
- *Project risks* affect schedule or resources;
- *Product risks* affect the quality or performance of the software being developed;
- *Business risks* affect the organisation developing or procuring the software.

Common project risks are derived from
- group/team itself (e.g. sickness, motivation, skills,...)
- customer (motivation, commitment, clearness of goals,...)
- technology (sw, hw, third party components,...)
- environment (other parties).

Examples of project, product, and business risks

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

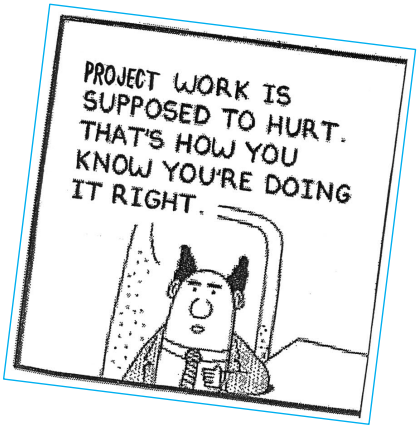
The risk management process

- **Risk identification**
  - Identify project, product and business risks;
- **Risk analysis**
  - Assess the likelihood and consequences of these risks;
- **Risk planning**
  - Draw up plans to avoid or minimise the effects of the risk;
- **Risk monitoring**
  - Monitor the risks throughout the project;

Risk identification

- May be a team activities or based on the individual project manager's **experience**.
- A checklist of common risks may be used to identify risks in a project
  - Technology risks.
  - Organizational risks.
  - People risks.
  - Requirements risks.
  - Estimation risks.

NO PAIN, NO GAIN



Examples of different risk types

Risk type	Possible risks
Estimation	The time required to develop the software is underestimated. (12) The rate of defect repair is underestimated. (13) The size of the software is underestimated. (14)
Organizational	The organization is restructured so that different management are responsible for the project. (6) Organizational financial problems force reductions in the project budget. (7)
People	It is impossible to recruit staff with the skills required. (3) Key staff are ill and unavailable at critical times. (4) Required training for staff is not available. (5)
Requirements	Changes to requirements that require major design rework are proposed. (10) Customers fail to understand the impact of requirements changes. (11)
Technology	The database used in the system cannot process as many transactions per second as expected. (1) Reusable software components contain defects that mean they cannot be reused as planned. (2)
Tools	The code generated by software code generation tools is inefficient. (8) Software tools cannot work together in an integrated way. (9)

Risk analysis

- ✧ Assess probability and seriousness of each risk.
- ✧ Probability may be very low, low, moderate, high or very high.
- ✧ Risk consequences might be catastrophic, serious, tolerable or insignificant.

Usually risks are calculated as

Risk probability * impact = seriousness.

It would be good if you could find some symptom or ”early warning sign” for every risk.

Risk types and examples

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget (7).	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project (3).	High	Catastrophic
Key staff are ill at critical times in the project (4).	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused. (2).	Moderate	Serious
Changes to requirements that require major design rework are proposed (10).	Moderate	Serious
The organization is restructured so that different management are responsible for the project (6).	High	Serious
The database used in the system cannot process as many transactions per second as expected (1).	Moderate	Serious

Risk types and examples

Risk	Probability	Effects
The time required to develop the software is underestimated (12).	High	Serious
Software tools cannot be integrated (9).	High	Tolerable
Customers fail to understand the impact of requirements changes (11).	Moderate	Tolerable
Required training for staff is not available (5).	Moderate	Tolerable
The rate of defect repair is underestimated (13).	Moderate	Tolerable
The size of the software is underestimated (14).	High	Tolerable
Code generated by code generation tools is inefficient (8).	Moderate	Insignificant

What-if questions

- What if several engineers are **ill** at the same time?
- What if an **economic downturn** leads to budget cuts of 20% for the project?
- What if the performance of **open-source software** is inadequate and the only expert on that open source software leaves?
- What if the company that **supplies** and maintains software components goes out of business?
- What if the **customer fails to deliver** the revised requirements as predicted?

Strategies to help manage risk, 1

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.

Strategies to help manage risk, 2

Risk	Strategy
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program generator.

Risk monitoring

- ✧ Assess each identified risks **regularly** to decide whether or not it is becoming less or more probable.
- ✧ Also assess whether the **effects** of the risk have changed.
- ✧ Each key risk should be discussed at management progress meetings.

It is NOT risk management if you write a risk list to Project Plan at the start of a project, and then forget those. Risk should be considered e.g. at every Sprint start meeting.

Risk indicators

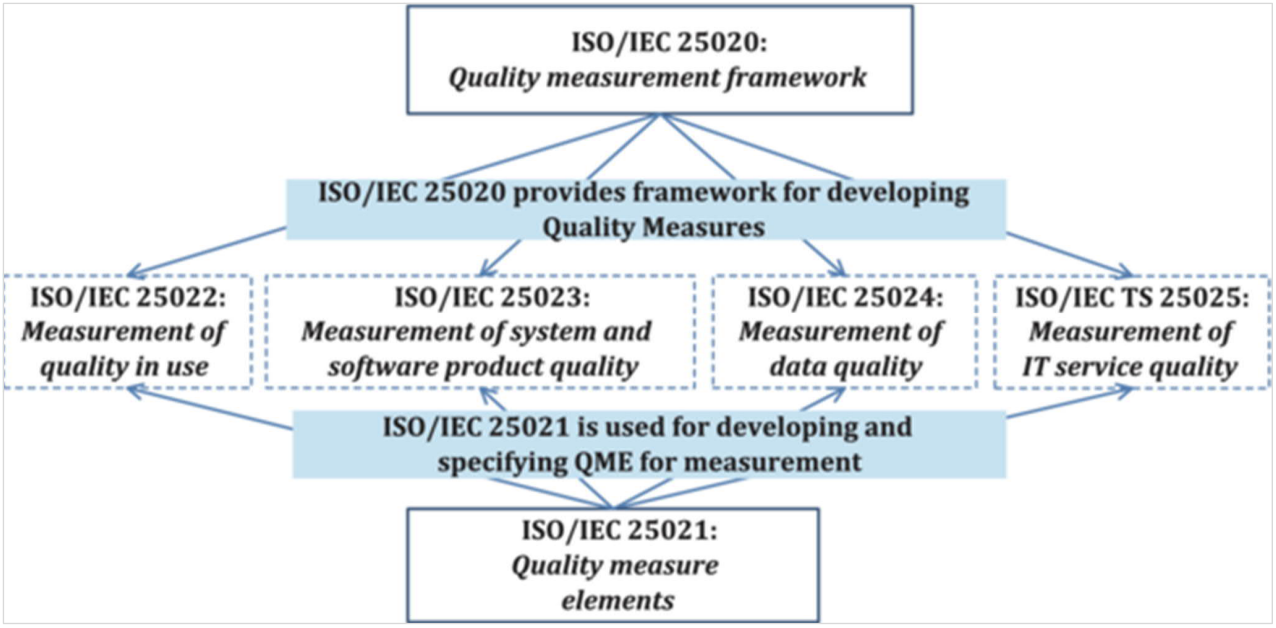
Risk type	Potential indicators
Estimation	Failure to meet agreed schedule; failure to clear reported defects.
Organizational	Organizational gossip; lack of action by senior management.
People	Poor staff morale; poor relationships amongst team members; high staff turnover.
Requirements	Many requirements change requests; customer complaints.
Technology	Late delivery of hardware or support software; many reported technology problems.
Tools	Reluctance by team members to use tools; complaints about CASE tools; demands for higher-powered workstations.

Tampereen yliopisto
Tampere University

Standards

Tampereen yliopisto
Tampere University

Quality standards



Tampereen yliopisto
Tampere University

Some standard...

STANDARDI

SFS-ISO 10006:2018

Suomen Standardisoimisliitto SFS ry
Finnish Standards Association SFS

Vahvistettu
2018-03-29

2. painos

1 (37)

SFS/ICS 03.100.70; 03.120.10

Korvaa standardin SFS-ISO 10006:en:2004

Replaces the standard SFS-ISO 10006:en:2004

Tämä standardi on vahvistettu englanninkielisenä.

This standard is approved in English.

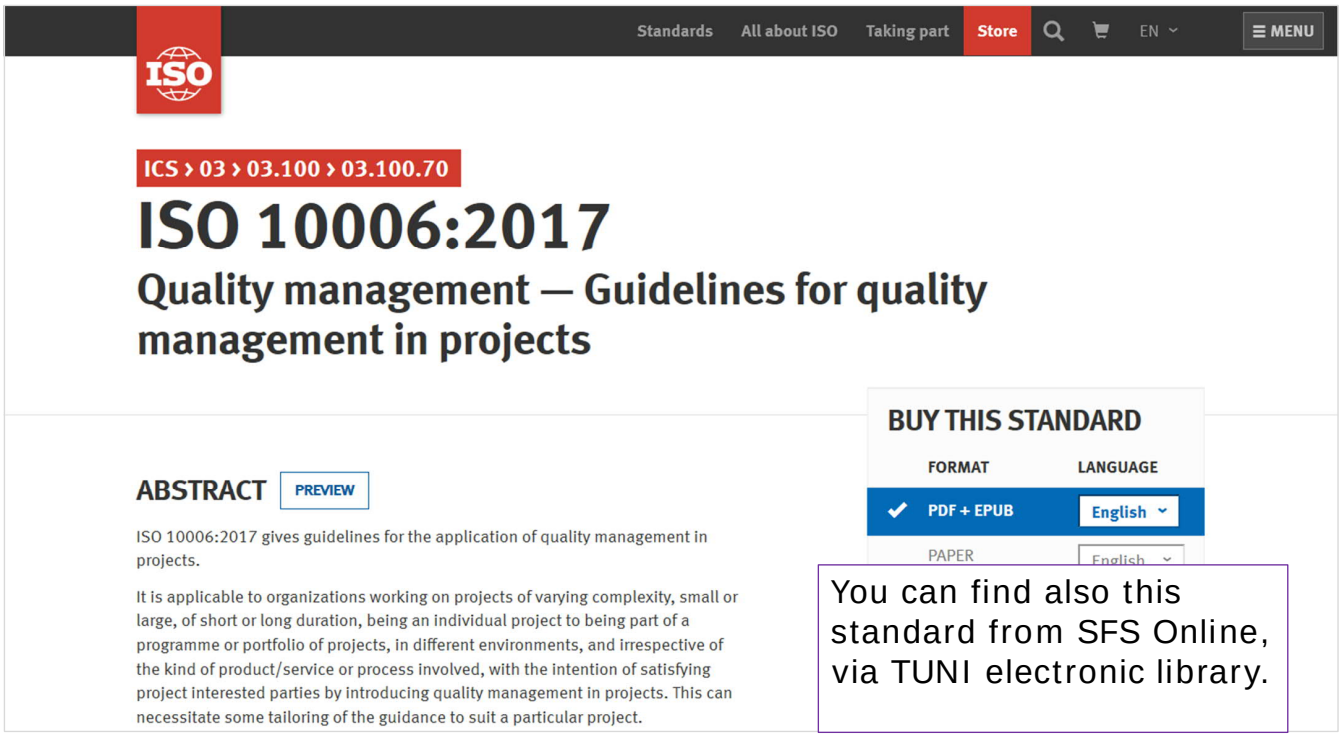
Quality management. Guidelines for quality management in projects

Tämä standardi sisältää kansainvälisen standardin ISO 10006:2017 "Quality management -- Guidelines for quality management in projects" englanninkielisen tekstin.

This standard consists of the English text of the International Standard ISO 10006:2017 "Quality management -- Guidelines for quality management in projects".

Kansainvälinen standardi ISO 10006:2017 on vahvistettu suomalaiseksi kansalliseksi standardiksi.

The International Standard ISO 10006:2017 has the status of a Finnish national standard.



You can find also this standard from SFS Online, via TUNI electronic library.

ISO 1006:2018

STANDARDI

SFS-ISO 10006:2018

Suomen Standardisoimisliitto SFS ry
Finnish Standards Association SFS

Vahvistettu
2018-03-29

2. painos

1 (37)

SFS/ICS 03.100.70; 03.120.10

Korvaa standardin SFS-ISO 10006:en:2004

Replaces the standard SFS-ISO 10006:en:2004

Tämä standardi on vahvistettu englanninkielisenä.

This standard is approved in English.

Quality management. Guidelines for quality management in projects

Tämä standardi sisältää kansainvälisen standardin ISO 10006:2017 "Quality management -- Guidelines for quality management in projects" englanninkielisen tekstin.

This standard consists of the English text of the International Standard ISO 10006:2017 "Quality management -- Guidelines for quality management in projects".

Kansainvälinen standardi ISO 10006:2017 on vahvistettu suomalaiseksi kansalliseksi standardiksi.

The International Standard ISO 10006:2017 has the status of a Finnish national standard.

Tampereen yliopisto
Tampere University

SQuaRE = Software Quality Requirements and Evaluation

Figure 1 (adapted from ISO/IEC 25000) illustrates the organization of the SQuaRE series representing families of standards, further called divisions.

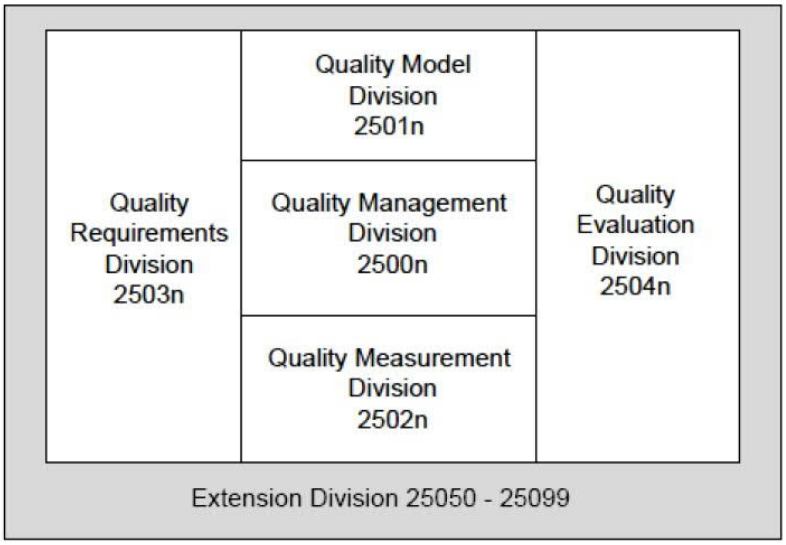
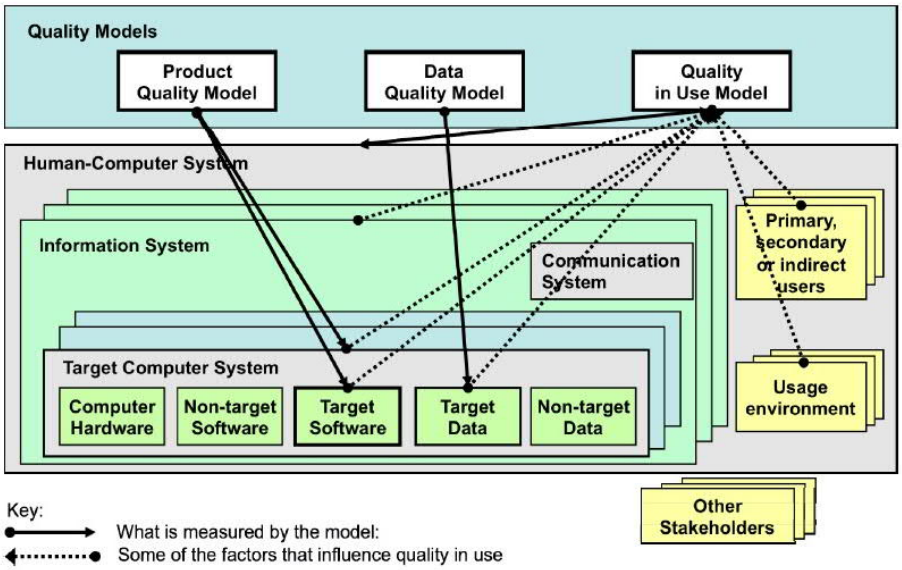


Figure 1 — Organization of SQuaRE series of International Standards

[ISO 25010:2011]

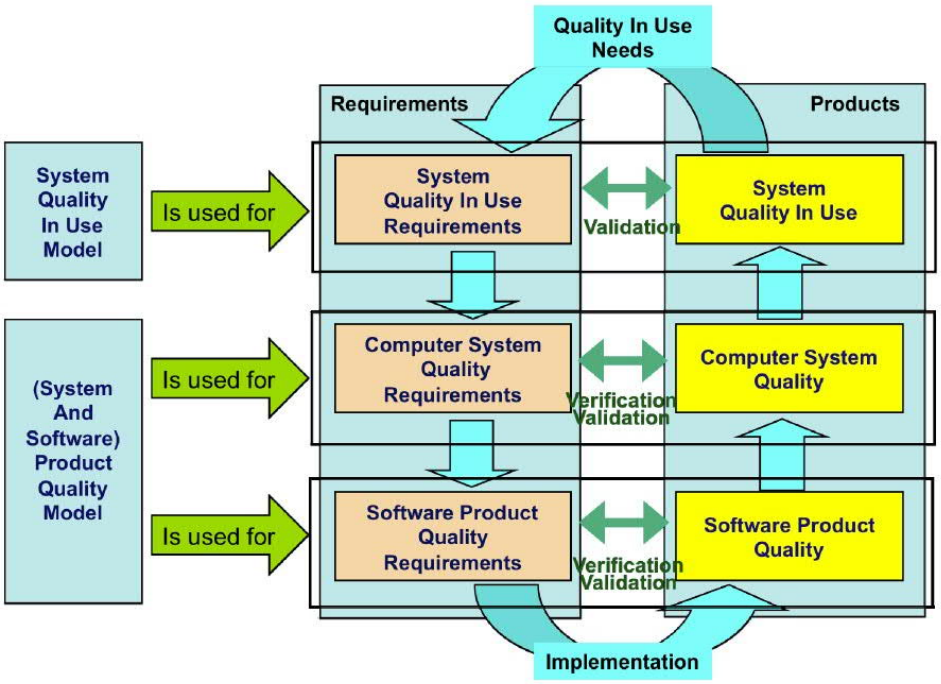
Tampereen yliopisto
Tampere University



NOTE This is conceptually the same as Figure 2 in ISO/IEC 25012 and Figure 5 in ISO/IEC 25030, but a different version that focuses on quality models.

Figure 5 — Targets of quality models

[ISO 25010:2011]



[ISO 25010:2011]

Figure C.5 — System/Software Quality Life Cycle Model

Finnish Software Measurement Association

FiSMA

Topic A: Quality measures of software product

- **TOP10 A.1 - Improvement of efficiency of end-user's work**
  - Type: Derived measure
  - Main content: A rate of user tasks, which are supported by the software compared to all other user tasks. A recommended method is a case study.
  - What the measure explains: How well and comprehensively the software fulfils user needs.
- **TOP10 A.2 - End-user satisfaction**
  - Type: Base measure
  - Main content : User experience. Could be divided to sub-measures. A recommended method is Net Promoter.
  - What the measure explains : How successfully the software serves the end-user e.g. usability and accessibility.

FISMA 2012 8

Finnish Software Measurement Association

FiSMA

Topic B: Software process

- **TOP10 B.1 - Maturity of the software process**
  - Type: Indicator, indirect measure
  - Main content: An operational level derived from a summary of selected processes. Well-known and widely suggested methods are CMMI and SPICE.
  - What the measure explains: Process wise capability of the supplier organisation to deliver products or services.
- **TOP10 B.2 - Agility of the software process**
  - Type: Indicator, indirect measure
  - Main content: A level of agility adaption with the whole software organisation. A recommended method is a survey or an employee inquiry.
  - What the measure explains: Ability to react to external changes or requests.
- **TOP10 B.3 - Improvability of the software process**
  - Type: Indicator, indirect measure
  - Main content: A rate of planned and decided improvement efforts which get completed accordingly. A recommended method is audit.
  - What the measure explains: Capability to execute while there is need to change and develop activities.

FiSMA 2012 9

Tampereen yliopisto
Tampere University

Finnish Software Measurement Association

FiSMA

Topic C: Management of a software project

- **TOP10 C.1 - Functional size of the software**
  - Type: Derived measure
  - Main content: A size of the software to be developed, acquired, maintained or which is the subject to other activity. A recommended method is FiSMA 1.1 or any other ISO/IEC-standard FSM method (e.g. function points, FP).
  - What the measure explains: Functional size enables comparisons of quality, efficiency and price data of systems of different sizes. Also a value of the software's functionality for the end-user.
- **TOP10 C.2 - Workload of the software project**
  - Type: Base measure
  - Main content: The complete workload of a defined development team in assigned activities during the life cycle of the system. A recommended unit of workload is an hour.
  - What the measure explains: Important source data for schedules, pricing and comparison of productivity.

FiSMA 2012 10

Finnish Software Measurement Association

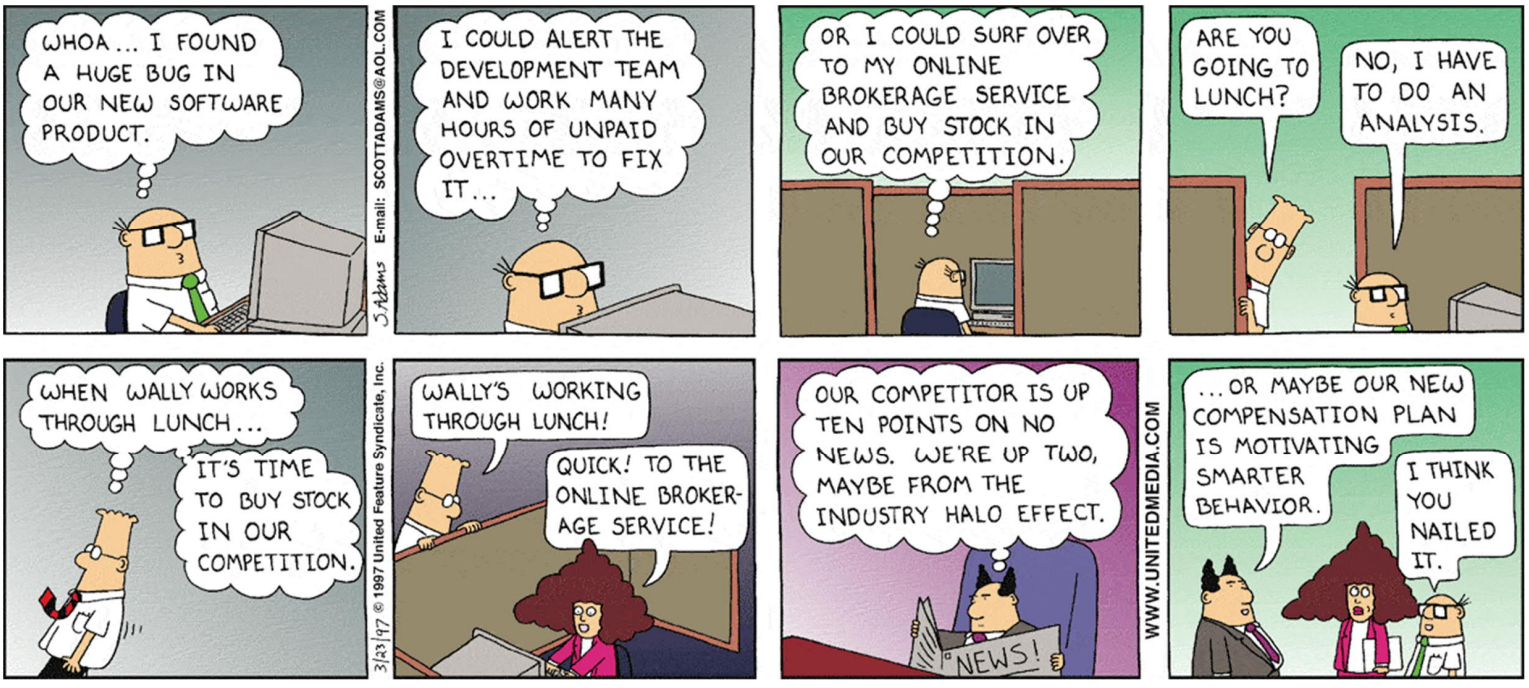
FiSMA

Topic D: Management of software business

- **TOP10 D.1 - Delivery speed of the software**
  - Type: Indicator, indirect measure
  - Main content: Functional size of the software delivered in the project divided by development time (FP/months).
  - What the measure explains: Delivery speed achieved in the project related to comparable ones; indicates competitiveness of both acquiring and supplying organisations.
- **TOP10 D.2 - Cost efficiency of the software purchase**
  - Type: Indicator, indirect measure
  - Main content: Total cost of the acquired software divided by a functional size, €/FP
  - What the measure explains: The cost efficiency of a project compared to similar ones; indicates competitiveness of both acquiring and supplying organisations.
- **TOP10 D.3 - Efficiency of the development portfolio**
  - Type: Derived measure, partly indicator, indirect measure
  - Main content: Revenues of a development portfolio compared to investments. A recommended method RoI or benefit/cost ratio.
  - What the measure explains: A competence to allocate and address IT efforts in accordance with business goals and value creation.

FiSMA 2012 11

Remember software project ethics



Just FYI: to use as an ”overkill” checklist.

EN 16602-80:2018 (E)

Software product assurance programme implementation	
5.1 Organization and responsibility	5.5 Procurement
5.2 Software product assurance programme management	5.6 Tools and supporting environment
5.3 Risk management and critical item control	5.7 Assessment and improvement process
5.4 Supplier selection and control	

Software process assurance
6.1 Software development life cycle
6.2 Requirements applicable to all software engineering processes
6.3 Requirements applicable to individual software engineering processes or activities

Software product quality assurance
7.1 Product quality objectives and metrication
7.2 Product quality requirements
7.3 Software intended for reuse
7.4 Standard ground hardware and services for operational system
7.5 Firmware

Figure 4-2: Structure of this Standard

EN 16602-80:2018 (E)

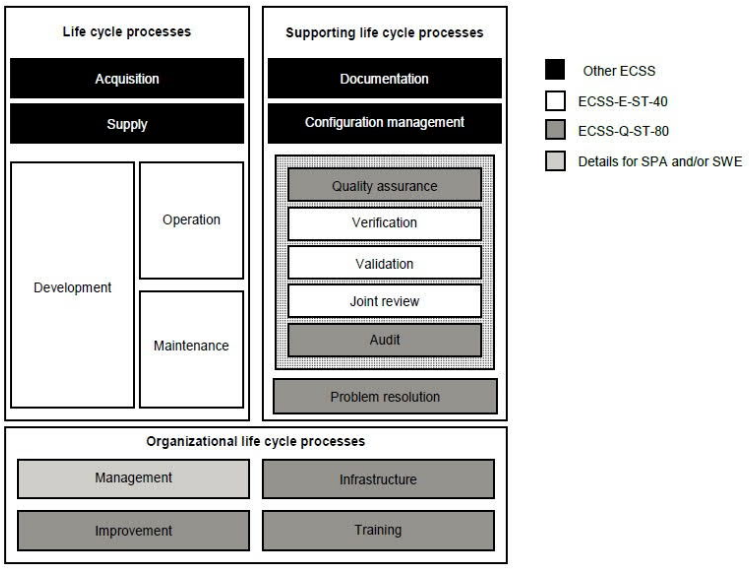


Figure 4-1: Software related processes in ECSS Standards

EN-16602-80:2018
Space product assurance. Software product assurance

Standardi

SFS-EN 16602-80:2018

Metalliteollisuuden Standardisoimisyhdistys ry
Mechanical Engineering and Metals Industry Standardization in Finland

Vahvistettu
2018-12-07

1 (1 + 117)

Tämä standardi on vahvistettu englanninkielisenä. *This standard is approved in English.*

Space product assurance. Software product assurance

Tämä standardi sisältää eurooppalaisen standardin EN 16602-80:2018 "Space product assurance. Software product assurance" englanninkielisen tekstin.

This standard consists of the English text of the European Standard EN 16602-80:2018 "Space product assurance. Software product assurance".

Eurooppalainen standardi EN 16602-80:2018 on vahvistettu suomalaiseksi kansalliseksi standardiksi.

The European Standard EN 16602-80:2018 has the status of a Finnish national standard.

This annex defines the structure of the software documents to be produced, as depicted in Figure A-1.

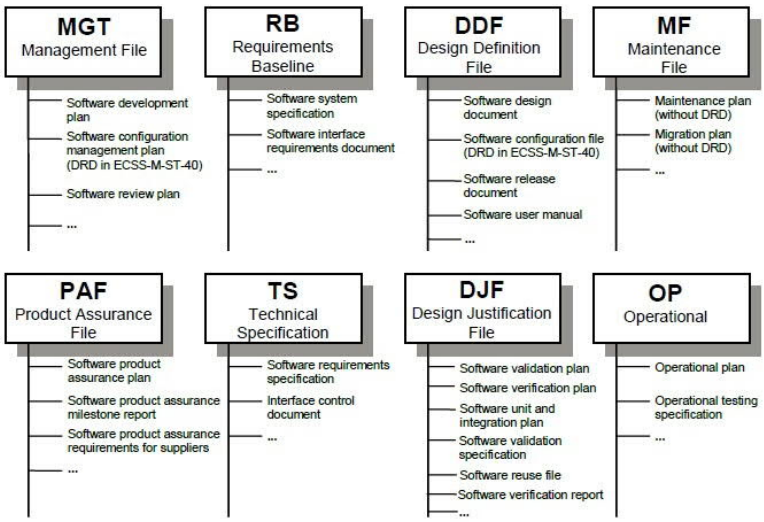


Figure A-1: Overview of software documents

Table A-1 represents the document requirements list, identifying the software documentation to be produced in accordance with the requirements defined in this Standard and in ECSS-E-ST-40.

Tampereen yliopisto
Tampere University

SFS-EN 16603-40: 2014 Space engineering - Part 40: Software

Processes needed in “heavy duty” software projects.

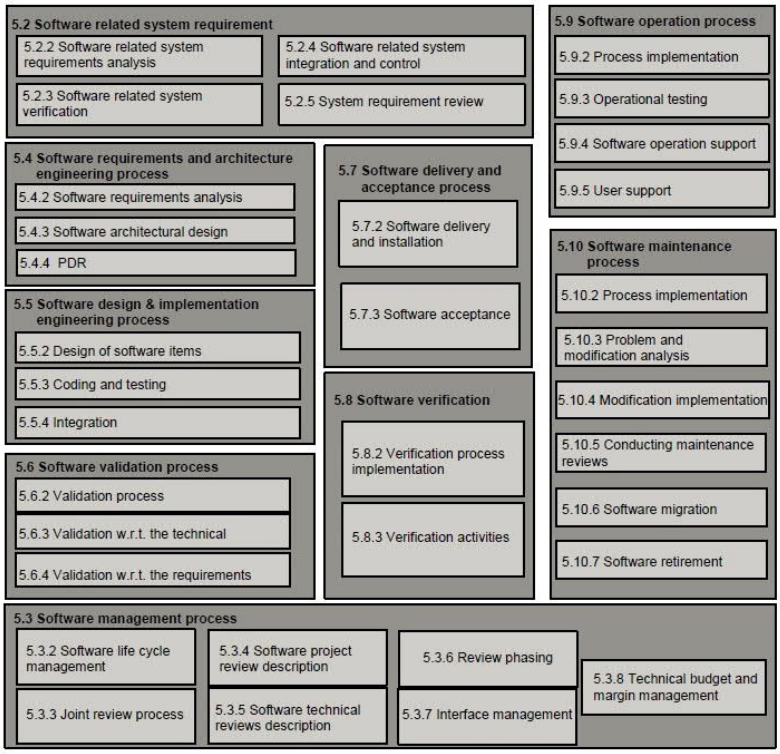


Figure 4-3: Structure of this Standard

Tampereen yliopisto
Tampere University

Security in standards

There are many standards about ”mission-critical” systems like Space systems or Health Care systems. You may look at those, for process guidance (start at TUNI electronic Library, SFS Online).

Today both cybersecurity and information security are also an issue in software development projects, which project manager should also think about (delegate if needed).

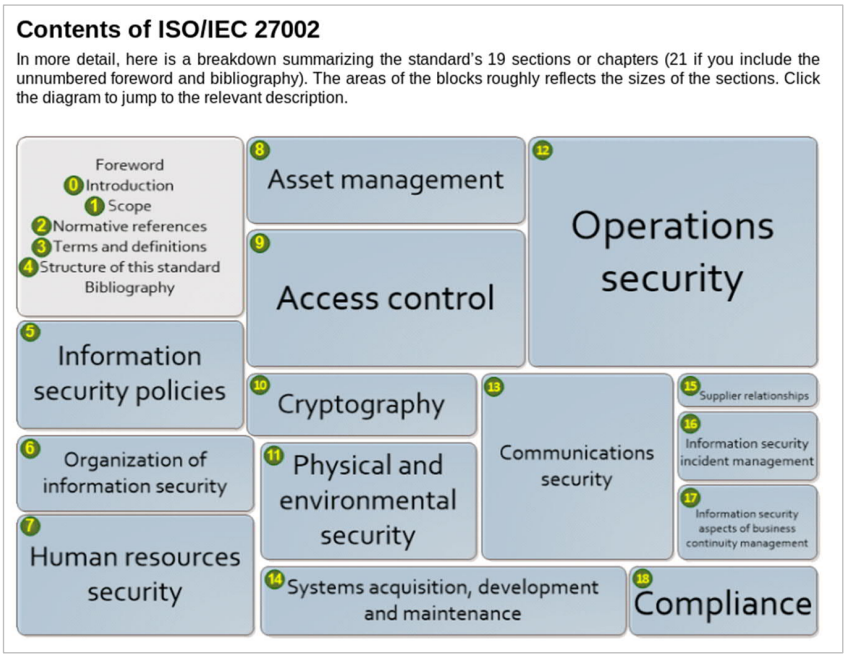
One collection www page is

https://www.iso27001security.com/html/iso27000.html

Tampereen yliopisto
Tampere University

ISO/IEC 27002:2013 — Information technology — Security techniques — **Code of practice for information security controls** *(second edition)*

Althoug these are more important in business, you may consider these also in software development projects.



Tampereen yliopisto
Tampere University

Project management environment

[ISO 21500:2012]

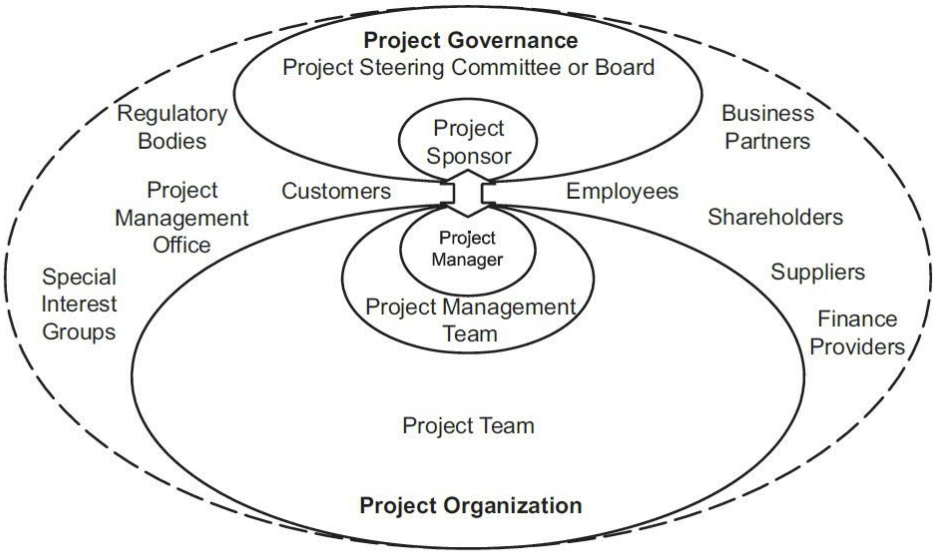
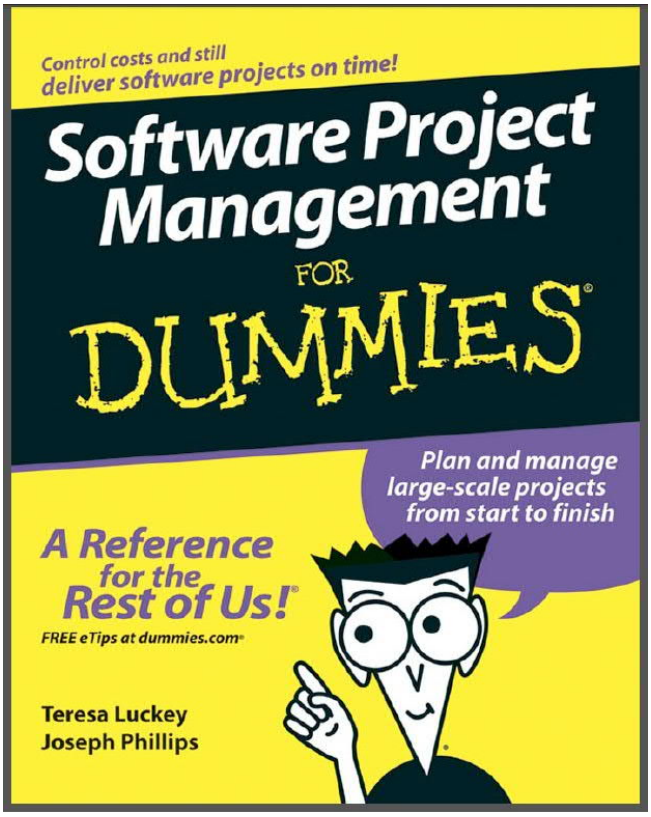
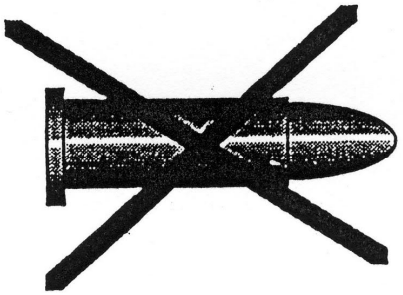


Figure 4 — Project stakeholders



There is no silver bullet!



No tool, no method will save you from having to think!

[sw eng folklore from 1980s]